4e Dimension

Langage
Windows®/Mac™OS

4e Dimension - Langage
Version 6.7 pour Windows® et Mac™ OS

Les informations contenues dans ce manuel peuvent faire l'objet de modifications sans préavis et ne sauraient en aucune manière engager 4D SA. La fourniture du logiciel décrit dans ce manuel est régie par un octroi de licence dont les termes sont précisés par ailleurs dans la licence électronique figurant sur le support du Logiciel et de la Documentation y afférente. Le logiciel et sa Documentation ne peuvent être utilisés, copiés ou reproduits sur quelque support que ce soit et de quelque manière que ce soit, que conformément aux termes de cette licence.

Aucune partie de ce manuel ne peut être reproduite ou recopiée de quelque manière que ce soit, électronique ou mécanique, y compris par photocopie, enregistrement, archivage ou tout autre procédé de stockage, de traitement et de récupération d'informations, pour d'autres buts que l'usage personnel de l'acheteur, et ce exclusivement aux conditions contractuelles, sans la permission explicite de 4D SA.

Sommaire

1. Introduction 31

Préface 33
Introduction 35
Construire une application 4D 46

2. Présentation du langage 61

Introduction au langage 4D 63
Types de données 69
Constantes 74
Variables 78
Variables système 84
Pointeurs 87
Nommer les objets du langage 4D 97
Conditions et boucles 108
Si...Sinon...Fin de si 110
Au cas ou...Sinon...Fin de cas 112
Tant que...Fin tant que 116
Repeter...Jusque 118
Boucle...Fin de boucle 119
Méthodes 125
Méthodes projet 131

3. BLOB 141

Commandes du thème BLOB 143
FIXER TAILLE BLOB 146
Taille BLOB 147
COMPRESSER BLOB 148
DECOMPRESSER BLOB 150
LIRE PROPRIETES BLOB 152
DOCUMENT VERS BLOB 154
BLOB VERS DOCUMENT 156
VARIABLE VERS BLOB 158

BLOB VERS VARIABLE 161
LISTE VERS BLOB 163
BLOB vers liste 165
ENTIER VERS BLOB 167
ENTIER LONG VERS BLOB 170
REEL VERS BLOB 173
TEXTE VERS BLOB 176
BLOB vers entier 179
BLOB vers entier long 181
BLOB vers reel 183
BLOB vers texte 185
INSERER DANS BLOB 188
SUPPRIMER DANS BLOB 189
COPIER BLOB 190
CRYPTER BLOB 191
DECRYPTER BLOB 196

4. Booléens 197

Commandes du thème Booléens 199
Vrai 200
Faux 201
Non 202

5. Chaînes de caractères 203

Chaine 205
Num 208
Position 210
Sous chaine 211
Longueur 213
Code ascii 214
Caractere 216
Symboles d'indice de chaîne 217
Majusc 220
Minusc 221
Remplacer caracteres 222

Inserer chaine 223
Supprimer chaine 224
Remplacer chaine 225
Mac vers Windows 227
Windows vers Mac 228
Mac vers ISO 229
ISO vers Mac 232

6. Commandes obsolètes 235

CHERCHER SUR CLE 237
TRIER SUR INDEX 238

7. Communications 239

REGLER SERIE 241
FIXER TIMEOUT 246
UTILISER FILTRE 248
ENVOYER PAQUET 250
RECEVOIR PAQUET 252
RECEVOIR BUFFER 255
ENVOYER VARIABLE 257
RECEVOIR VARIABLE 258
ENVOYER ENREGISTREMENT 259
RECEVOIR ENREGISTREMENT 260

8. Compilateur 265

Commandes du thème Compilateur 267
C_BLOB 271
C_BOOLEEN 272
C_DATE 273
C_GRAPHE 274
C_ENTIER 275
C_ENTIER LONG 276
C_IMAGE 277

C_POINTEUR 278
C_REEL 279
C_ALPHA 280
C_TEXTE 281
C_HEURE 282
APPELER 4D 283

9. Dates et heures 285

Date du jour 287
Jour de 290
Mois de 291
Annee de 293
Numero du jour 294
Ajouter a date 296
Date 297
Heure courante 298
Chaine heure 299
Heure 300
Nombre de ticks 301
Nombre de millisecondes 302
SIECLE PAR DEFAUT 303

10. Débogueur 305

Un débogueur, pour quoi faire ? 307
Fenêtre d'erreur de syntaxe 312
Débogueur 314
Fenêtre d'expression 320
Fenêtre de chaîne d'appel 326
Fenêtre d'évaluation 328
Fenêtre d'évaluation des méthodes 331
Points d'arrêt 335
Liste des points d'arrêt 338
Points d'arrêt sur commandes 341
Raccourcis du débogueur 345

11. Définition structure 347

Commandes du thème Définition structure 349
Nombre de tables 350
Nombre de champs 351
Nom de la table 352
Nom du champ 353
Table 354
LIRE PROPRIETES TABLE 355
Champ 356
LIRE PROPRIETES CHAMP 357
LIRE PROPRIETES SAISIE CHAMP 359
LIRE PROPRIETES LIEN 360
FIXER INDEX 361
Lire parametre base 363
FIXER PARAMETRE BASE 365

12. Documents système 371

Présentation des documents système 373
Type document 379
CHANGER TYPE DOCUMENT 380
Createur document 381
CHANGER CREATEUR DOCUMENT 382
Ouvrir document 383
Creer document 386
Ajouter a document 388
FERMER DOCUMENT 389
COPIER DOCUMENT 390
DEPLACER DOCUMENT 391
SUPPRIMER DOCUMENT 392
Tester chemin acces 393
CREER DOSSIER 394
Selectionner dossier 395

SUPPRIMER DOSSIER 398
CREER ALIAS 399
RESOUDRE ALIAS 401
LISTE DES VOLUMES 402
PROPRIETES DU VOLUME 403
LISTE DES DOSSIERS 406
LISTE DES DOCUMENTS 407
ASSOCIER TYPES FICHIER 408
PROPRIETES DOCUMENT 410
CHANGER PROPRIETES DOCUMENT 416
LIRE ICONE DOCUMENT 417
Taille document 418
CHANGER TAILLE DOCUMENT 419
Position dans document 420
CHANGER POSITION DANS DOCUMENT 421

13. Enregistrements 423

AFFICHER ENREGISTREMENT 425
CREER ENREGISTREMENT 426
DUPLIQUER ENREGISTREMENT 427
Nouvel enregistrement 428
Enregistrement modifie 429
Enregistrement charge 430
STOCKER ENREGISTREMENT 431
SUPPRIMER ENREGISTREMENT 433
Enregistrements dans table 434
Numero enregistrement 435
ALLER A ENREGISTREMENT 436
Numerotation automatique 437
A propos des numéros d'enregistrements 439
EMPILER ENREGISTREMENT 442
DEPILER ENREGISTREMENT 443
Utiliser la pile d'enregistrements 444

14. Enregistrements (verrouillage) 445

Verrouillage d'enregistrements 447
LECTURE ECRITURE 454
LECTURE SEULEMENT 455
Etat lecture seulement 456
CHARGER ENREGISTREMENT 457
LIBERER ENREGISTREMENT 458
Enregistrement verrouille 459
VERROUILLE PAR 460

15. Ensembles 461

Présentation des ensembles 463
ENSEMBLE VIDE 468
NOMMER ENSEMBLE 469
CREER ENSEMBLE SUR TABLEAU 470
UTILISER ENSEMBLE 471
ADJOINDRE ELEMENT 472
ENLEVER ELEMENT 473
EFFACER ENSEMBLE 474
Appartient a ensemble 475
Enregistrements dans ensemble 476
STOCKER ENSEMBLE 477
CHARGER ENSEMBLE 479
DIFFERENCE 481
INTERSECTION 483
REUNION 485
COPIER ENSEMBLE 487

16. Environnement 4D 489

Type application 491
Type version 492
Version application 493
Application compilee 495
PROPRIETES PLATE FORME 496
Fichier application 500
Fichier structure 501
Fichier donnees 503
Dossier 4D 505
LISTE SEGMENTS DE DONNEES 507
AJOUTER SEGMENT DE DONNEES 509
ECRIRE CACHE 510
QUITTER 4D 511
FIXER HISTORIQUE 513
LIRE INFORMATIONS SERIALISATION 515

17. Environnement système 517

Hauteur ecran 519
Largeur ecran 520
Nombre ecrans 521
COORDONNEES ECRAN 522
PROFONDEUR ECRAN 523
FIXER PROFONDEUR ECRAN 525
Ecran principal 526
Hauteur barre de menus 527
LISTE DES POLICES 528
Nom de police 529
Numero de police 530
Dossier systeme 531
Dossier temporaire 533

Nom de la machine 534
Nom du possesseur 535
Gestalt 536
ENREGISTRER EVENEMENT 537

18. Evénements formulaire 539

Evenement formulaire 541
Avant 559
Pendant 560
Apres 561
En entete 562
En rupture 563
En pied 564
Activation 565
Desactivation 566
Appel exterieur 567
Lire texte edite 568
FIXER MINUTEUR 570

19. Fenêtres 573

Gestion des fenêtres 575
Types de fenêtres 576
Creer fenetre 582
Creer fenetre externe 586
FERMER FENETRE 588
EFFACER FENETRE 589
REDESSINER FENETRE 590
DEPLACER FENETRE 591
CACHER FENETRE 593
AFFICHER FENETRE 594
MAXIMISER FENETRE 595
MINIMISER FENETRE 597
Titre fenetre 598
CHANGER TITRE FENETRE 599

CACHER BARRE OUTILS 601
AFFICHER BARRE OUTILS 602
LISTE FENETRES 603
Type fenetre 604
Process de la fenetre 605
COORDONNEES FENETRE 606
CHANGER COORDONNEES FENETRE 607
Fenetre premier plan 608
Fenetre suivante 609
Chercher fenetre 610
Creer fenetre formulaire 611
LIRE PROPRIETES FORMULAIRE 613

20. Fonctions mathématiques 615

Abs 617
Ent 618
Dec 619
Arrondi 620
Troncature 621
Hasard 622
Modulo 623
Racine carree 624
Log 625
Exp 626
Sin 627
Cos 628
Tan 629
Arctan 630
FIXER NIVEAU COMPARAISON REEL 631
Affichage des nombres réels 632
Convertisseur Euro 634

21. Fonctions statistiques 637

Présentation des fonctions statistiques 639
Somme 640
Moyenne 641
Min 642
Max 643
Ecart type 644
Variance 645
Somme des carres 647

22. Gestion de la saisie 649

VALIDER 651
NE PAS VALIDER 652
Frappe clavier 653
FILTRER FRAPPE CLAVIER 658
ALLER A CHAMP 664
REFUSER 665

23. Glisser-Déposer 667

Présentation du Glisser-Déposer 669
Position deposer 678
PROPRIETES GLISSER DEPOSER 679

24. Graphes 687

GRAPHE 689
PARAMETRES DU GRAPHE 694
GRAPHE SUR SELECTION 696

25. Images 699

Introduction aux images 701
COMPRESSER IMAGE 704
CHARGER ET COMPRESSER IMAGE 705
COMPRESSER FICHIER IMAGE 707
ENREGISTRER IMAGE 708
IMAGE VERS GIF 709
IMAGE VERS BLOB 711
BLOB VERS IMAGE 712
ECRIRE FICHIER IMAGE 713
LIRE FICHIER IMAGE 714
LISTE TYPES IMAGES 715
Taille image 717
PROPRIETES IMAGE 718
CREER IMAGETTE 719
LISTE IMAGES DANS BIBLIOTHEQUE 721
LIRE IMAGE DANS BIBLIOTHEQUE 723
ECRIRE IMAGE DANS BIBLIOTHEQUE 725
SUPPRIMER IMAGE DANS BIBLIOTHEQUE 728

26. Import-Export 729

LECTURE ASCII 731
ECRITURE ASCII 733
LECTURE SYLK 735
ECRITURE SYLK 737
LECTURE DIF 739
ECRITURE DIF 741
IMPORTER DONNEES 743
EXPORTER DONNEES 745

27. Impressions 747

ETAT 749
IMPRIMER ETIQUETTES 751
IMPRIMER SELECTION 754
Page impression 756
NIVEAUX DE RUPTURES 757
CUMULER SUR 758
Sous total 759
Niveau 762
IMPRIMER ENREGISTREMENT 764
UTILISER PARAMETRES IMPRESSION 766
PARAMETRES IMPRESSION 768
IMPRESSION ECRAN 769
IMPRIMER LIGNE 770
SAUT DE PAGE 772

28. Interface utilisateur 773

BEEP 775
JOUER SON 776
Lire interface 778
FIXER INTERFACE 779
FIXER TITRES TABLES 781
FIXER TITRES CHAMPS 785
Majuscule enfoncee 787
Verrouillage majuscule enfoncee 788
Windows Ctrl enfoncee 789
Windows Alt enfoncee 790
Macintosh commande enfoncee 791
Macintosh option enfoncee 792
Macintosh control enfoncee 793
POSITION SOURIS 794
Pop up menu 795
GENERER FRAPPE CLAVIER 798
GENERER CLIC 799
GENERER EVENEMENT 800

TEXTE SELECTIONNE 802
SELECTIONNER TEXTE 803
CHANGER POINTEUR SOURIS 804
Dernier objet 805
REDESSINER 806
INVERSER FOND 807

29. Interruptions 809

APPELER SUR EVENEMENT 811
FILTRER EVENEMENT 815
APPELER SUR ERREUR 816
STOP 820

30. Langage 823

Nombre de parametres 825
Type 827
Self 830
RESOUDRE POINTEUR 831
Nil 833
Est une variable 834
Pointeur vers 835
EXECUTER 836
Nom commande 837
Nom methode courante 840
TRACE 841
PAS DE TRACE 843

31. Liens 845

Présentation des liens 847
LIENS AUTOMATIQUES 850
CHARGER SUR LIEN 851
LIEN RETOUR 855
CREER SUR LIEN 858
STOCKER SUR LIEN 859
CHARGER ANCIEN 860
STOCKER ANCIEN 861
ANCIEN LIEN RETOUR 862
JOINTURE 863
SELECTION RETOUR 864

32. Listes hiérarchiques 865

Charger liste 867
STOCKER LISTE 869
Nouvelle liste 870
Copier liste 871
SUPPRIMER LISTE 872
Nombre elements 874
Liste existante 876
REDESSINER LISTE 877
CHANGER PROPRIETES LISTE 878
LIRE PROPRIETES LISTE 887
TRIER LISTE 889
AJOUTER A LISTE 891
INSERER ELEMENT 899
CHANGER PROPRIETES ELEMENT 900
LIRE PROPRIETES ELEMENT 902
Position element liste 903
Element parent 904

SUPPRIMER ELEMENT 906
INFORMATION ELEMENT 908
CHANGER ELEMENT 910
Element selectionne 912
SELECTIONNER ELEMENT 914
SELECTIONNER ELEMENT PAR REFERENCE 916

33. Menus 917

Gestion des menus 919
CHANGER BARRE 923
CACHER BARRE DE MENUS 925
AFFICHER BARRE DE MENUS 926
APPELER SUR A PROPOS 927
Menu choisi 928
Nombre de menus 930
Nombre de lignes du menu 931
Titre menu 932
Texte ligne menu 933
CHANGER TEXTE LIGNE MENU 934
Style ligne menu 935
CHANGER STYLE LIGNE MENU 937
Marque ligne menu 938
MARQUER LIGNE MENU 939
Raccourci clavier 940
CHANGER RACCOURCI CLAVIER 941
INACTIVER LIGNE MENU 942
ACTIVER LIGNE MENU 943
AJOUTER LIGNE MENU 944
INSERER LIGNE MENU 946
SUPPRIMER LIGNE MENU 947

34. Messages 949

SUPPRIMER MESSAGES 951
LAISSER MESSAGES 952
ALERTE 953
CONFIRMER 955
Demander 958
MESSAGE 961
POSITION MESSAGE 966

35. Méthodes base 969

Présentation des méthodes base 971
Méthode base Sur ouverture 973
Méthode base Sur fermeture 975

36. Opérateurs 981

Opérateurs 983
Opérateurs sur les chaînes 985
Opérateurs numériques 986
Opérateurs sur les dates 987
Opérateurs sur les heures 988
Opérateurs de comparaison 990
Opérateurs logiques 995
Opérateurs sur les images 997
Opérateurs sur les bits 1006

37. Pages formulaire 1011

Commandes du thème Pages formulaire 1013
PREMIERE PAGE 1014
DERNIERE PAGE 1015
PAGE SUIVANTE 1016
PAGE PRECEDENTE 1017
ALLER A PAGE 1018
Page formulaire courante 1019

38. Presse-Papiers 1021

AJOUTER A PRESSE PAPIERS 1023
EFFACER PRESSE PAPIERS 1029
LIRE PRESSE PAPIERS 1030
LIRE IMAGE DANS PRESSE PAPIERS 1032
Lire texte dans presse papiers 1033
ECRIRE IMAGE DANS PRESSE PAPIERS 1035
ECRIRE TEXTE DANS PRESSE PAPIERS 1036
Tester presse papiers 1037

39. Process 1039

Introduction aux process 1041
Nouveau process 1045
Executer sur serveur 1048
ENDORMIR PROCESS 1053
SUSPENDRE PROCESS 1054
REACTIVER PROCESS 1055
Process interrompu 1056
Numero du process courant 1057
Statut du process 1058
INFORMATIONS PROCESS 1060

Chercher process 1063
Nombre utilisateurs 1065
Nombre de process 1066
Nombre de process utilisateurs 1067
EXECUTER SUR CLIENT 1068
INSCRIRE CLIENT 1070
DESINSCRIRE CLIENT 1073
LIRE CLIENTS INSCRITS 1074

40. Process (Communications) 1075

Semaphore 1077
EFFACER SEMAPHORE 1080
Tester semaphore 1081
APPELER PROCESS 1082
LIRE VARIABLE PROCESS 1084
ECRIRE VARIABLE PROCESS 1087
VARIABLE VERS VARIABLE 1090

41. Process (Interface utilisateur) 1093

CACHER PROCESS 1095
MONTRER PROCESS 1096
PASSER AU PREMIER PLAN 1097
Process de premier plan 1098

42. Propriétés des objets 1099

Propriétés des objets 1101
CHANGER JEU DE CARACTERES 1103
CHANGER TAILLE 1104
CHANGER STYLE 1105
ACTIVER BOUTON 1107
INACTIVER BOUTON 1109
TITRE BOUTON 1111

CHOIX FORMATAGE 1113
CHOIX FILTRE SAISIE 1116
CHOIX ENUMERATION 1118
CHOIX SAISISSABLE 1119
CHOIX VISIBLE 1121
CHOIX COULEUR 1123
FIXER COULEURS RVB 1125
LIRE RECT OBJET 1129
DEPLACER OBJET 1130

43. Protocole sécurisé 1133

GENERER CLES CRYPTAGE 1135
GENERER DEMANDE CERTIFICAT 1137

44. Recherches et tris 1141

CHERCHER PAR EXEMPLE 1143
CHERCHER 1144
CHERCHER DANS SELECTION 1151
CHERCHER PAR FORMULE 1153
CHERCHER PAR FORMULE DANS SELECTION 1155
CHERCHER PAR TABLEAU 1156
FIXER DESTINATION RECHERCHE 1157
FIXER LIMITE RECHERCHE 1163
Trouver clef index 1165
TRIER 1166
TRIER PAR FORMULE 1171

45. Ressources 1173

Ressources 1175
Les ressources et 4D Insider : un exemple 1183
Ouvrir fichier ressources 1190
Creer fichier ressources 1194
FERMER FICHIER RESSOURCES 1196
LISTE TYPES RESSOURCE 1197
LISTE RESSOURCES 1199
LISTE DE CHAINES VERS TABLEAU 1201
TABLEAU VERS LISTE DE CHAINES 1202
Lire chaine dans liste 1204
Lire ressource chaine 1205
ECRIRE RESSOURCE CHAINE 1206
Lire ressource texte 1207
ECRIRE RESSOURCE TEXTE 1208
LIRE RESSOURCE IMAGE 1209
ECRIRE RESSOURCE IMAGE 1210
LIRE RESSOURCE ICONE 1211
LIRE RESSOURCE 1213
ECRIRE RESSOURCE 1215
Lire nom ressource 1218
ECRIRE NOM RESSOURCE 1220
Lire proprietes ressource 1221
ECRIRE PROPRIETES RESSOURCE 1222
SUPPRIMER RESSOURCE 1225
Lire ID ressource composant 1227

46. Saisie 1229

AJOUTER ENREGISTREMENT 1231
MODIFIER ENREGISTREMENT 1234
AJOUTER SOUS ENREGISTREMENT 1236
MODIFIER SOUS ENREGISTREMENT 1238
DIALOGUE 1239
Modifie 1241
Ancien 1243

47. Sélections 1245

TOUT SELECTIONNER 1247
Enregistrements trouves 1248
SUPPRIMER SELECTION 1249
Numero dans selection 1251
ALLER DANS SELECTION 1252
DEBUT SELECTION 1254
ENREGISTREMENT SUIVANT 1255
ALLER A DERNIER ENREGISTREMENT 1256
ENREGISTREMENT PRECEDENT 1257
Avant selection 1258
Fin de selection 1260
VISUALISER SELECTION 1262
MODIFIER SELECTION 1266
APPLIQUER A SELECTION 1267
REDUIRE SELECTION 1269
SCAN INDEX 1271
ENREGISTREMENT SELECTION 1272
MARQUER ENREGISTREMENTS 1273

48. Sélections Temporaires 1275

Présentation des Sélections Temporaires 1277
COPIER SELECTION 1279
DEPLACER SELECTION 1281
UTILISER SELECTION 1282
EFFACER SELECTION 1283
CREER SELECTION SUR TABLEAU 1284

49. Serveur Web 1285

Services Web, Présentation 1287
Services Web, Configuration 1291
Services Web, Premiers pas (Partie I) 1298
Services Web, Premiers pas (Partie II) 1305
Services Web, Utiliser le protocole SSL 1312
Services Web, Sécurité des connexions 1316
Méthode base Sur authentification Web 1323
Services Web, Process de connexion Web 1327
Méthode base Sur connexion Web 1337
Services Web, Paramétrages du Serveur Web 1345
Services Web, Mode sans contexte 1352
Services Web, Support du HTML 1357
Services Web, Support de XML et WML 1365
Encapsulation HTML et Javascript 1366
Services Web, URLs et actions de formulaires 1380
Services Web, Balises HTML 4D 1389
Services Web, Informations sur le site Web 1398
Services Web, Support des CGI 1402
LANCER SERVEUR WEB 1410
ARRETER SERVEUR WEB 1411
FIXER TEMPORISATION WEB 1412
FIXER RACINE HTML 1413
FIXER LIMITES AFFICHAGE WEB 1414
FIXER PAGE ACCUEIL 1417
ENVOYER FICHIER HTML 1418
ENVOYER BLOB HTML 1422
ENVOYER TEXTE HTML 1425
LIRE VARIABLES FORMULAIRE WEB 1426
Contexte Web 1428
FIXER ENTETE HTTP 1429
LIRE ENTETE HTTP 1431
ENVOYER REDIRECTION HTTP 1434
STATISTIQUES DU CACHE WEB 1436
Connexion Web securisee 1437
OUVRIR URL WEB 1438

50. Sous-enregistrements 1439

CREER SOUS ENREGISTREMENT 1441
SUPPRIMER SOUS ENREGISTREMENT 1442
TOUS LES SOUS ENREGISTREMENTS 1444
Sous enregistrements trouves 1445
APPLIQUER A SOUS SELECTION 1446
DEBUT SOUS ENREGISTREMENT 1447
ALLER A DERNIER SOUS ENREGISTREMENT 1448
SOUS ENREGISTREMENT SUIVANT 1449
SOUS ENREGISTREMENT PRECEDENT 1450
Avant sous enregistrement 1451
Fin sous enregistrement 1452
TRIER SOUS ENREGISTREMENTS 1453
CHERCHER SOUS ENREGISTREMENTS 1454

51. Table 1455

TABLE PAR DEFAUT 1457
Table par defaut courante 1459
FORMULAIRE ENTREE 1460
FORMULAIRE SORTIE 1462
Table du formulaire courant 1464

52. Tableaux 1467

Présentation des tableaux 1469
Créer des tableaux 1470
Tableaux et objets de formulaire 1473
Zones de défilement groupées 1482
Les tableaux et le langage 4D 1485
Tableaux et pointeurs 1487
Utiliser l'élément zéro d'un tableau 1489
Tableaux à deux dimensions 1491
Tableaux et mémoire 1493

TABLEAU ENTIER 1496
TABLEAU ENTIER LONG 1498
TABLEAU REEL 1500
TABLEAU ALPHA 1502
TABLEAU TEXTE 1504
TABLEAU DATE 1506
TABLEAU BOOLEEN 1508
TABLEAU IMAGE 1510
TABLEAU POINTEUR 1512
Taille tableau 1514
TRIER TABLEAU 1515
Chercher dans tableau 1517
INSERER LIGNES 1519
SUPPRIMER LIGNES 1520
COPIER TABLEAU 1521
ENUMERATION VERS TABLEAU 1522
TABLEAU VERS ENUMERATION 1524
SELECTION VERS TABLEAU 1526
SELECTION LIMITEE VERS TABLEAU 1528
TABLEAU VERS SELECTION 1531
VALEURS DISTINCTES 1533
TABLEAU ENTIER LONG SUR SELECTION 1535
TABLEAU BOOLEEN SUR ENSEMBLE 1536

53. Transactions 1537

Utiliser des transactions 1539
DEBUT TRANSACTION 1544
VALIDER TRANSACTION 1545
ANNULER TRANSACTION 1546
Transaction en cours 1547

54. Triggers 1549

Présentation des triggers 1551
Evenement moteur 1562
Niveau du trigger 1564
PROPRIETES DU TRIGGER 1565

55. Utilisateurs et groupes 1567

CHANGER PRIVILEGES 1569
CHANGER UTILISATEUR 1570
Valider mot de passe 1571
CHANGER MOT DE PASSE 1572
Utilisateur courant 1573
Appartient au groupe 1574
SUPPRIMER UTILISATEUR 1575
Utilisateur supprime 1576
LIRE LISTE UTILISATEURS 1577
LIRE PROPRIETES UTILISATEUR 1579
Ecrire proprietes utilisateur 1581
LIRE LISTE GROUPE 1584
LIRE PROPRIETES GROUPE 1585
Ecrire proprietes groupe 1587
CHANGER LICENCES 1590

56. Variables 1593

ECRIRE VARIABLES 1595
LIRE VARIABLES 1597
EFFACER VARIABLE 1598
Indefinie 1600

57. Codes d'erreurs 1603

Erreurs de syntaxe 1605
Erreurs de la base de données 1608
Erreurs des composants réseaux 1611
Erreurs du gestionnaire de fichiers du système 1612
Erreurs du gestionnaire de mémoire du système 1614
Erreurs du gestionnaire d'impression du système 1615
Erreurs du gestionnaire de ressources du système 1616

Erreurs SANE NaN 1617
Erreurs du gestionnaire de son du système 1618
Erreurs du gestionnaire de port série du système 1619
Erreurs MacOS 1620
Tester le verrouillage du fichier de données 1621

58. Codes ASCII 1625

Codes ASCII 1627
Codes ASCII 0..63 1629
Codes ASCII 64..127 1631
Codes ASCII 128..191 1633
Codes ASCII 192..255 1637
Codes des touches de fonction 1641

59. Syntaxe des commandes 1643

Syntaxe des commandes (liste alphabétique) 1645

Constantes 1661

BLOB 1663
Caractères latins ISO 1664
Chercher fenetre 1666
Codes ASCII 1667
Communications 1669
Couleurs 1670
Creer fenetre 1671
Destinations de recherche 1672
Documents système 1673
Dossier Système 1674

Environnement 4D ... 1675
Euro monnaies ... 1676
Evénements (Modifiers) ... 1677
Evénements (types) ... 1678
Evénements formulaire ... 1679
Evénements moteur ... 1680
Expressions ... 1681
FIXER COULEUR RVB ... 1682
Fonctions mathématiques ... 1683
Formats d'affichage des dates ... 1684
Formats d'affichage des heures ... 1685
Formats d'affichage des images ... 1686
Interface de plate-forme ... 1687
Journal d'événements Windows NT ... 1688
Jours et mois ... 1689
Listes hiérarchiques ... 1690
Moteur de la base ... 1691
Numéros de port TCP ... 1692
Paramètres de la base ... 1693
Position fenêtre ... 1694
Presse-Papiers ... 1695
PROFONDEUR ECRAN ... 1696
Propriétés des ressources ... 1697
Propriétés plate-forme ... 1698
Signatures système standard ... 1699
Statut du process ... 1700
Styles de caractères ... 1701
Touches de fonction ... 1702
Type de process ... 1703
Type fenetre ... 1704
Types champs et variables ... 1705

Index des commandes ... 1707

1

Introduction

4e Dimension possède son propre langage de programmation. Ce langage intégré, qui comprend plus de 500 commandes, fait de 4e Dimension un outil de développement très puissant.

Le langage de 4e Dimension peut être utilisé pour de multiples types de tâches, de la réalisation de calculs simples à la création d'interfaces utilisateur personnalisées complexes. Par l'intermédiaire des routines du langage, vous pourrez par exemple :

- Accéder par programmation à tous les éditeurs du mode Utilisation,
- Créer et imprimer des états et des étiquettes complexes avec les données de la base,
- Communiquer avec d'autres systèmes d'information,
- Gérer des documents,
- Importer et exporter des données entre des bases 4e Dimension et d'autres applications,
- Incorporer des procédures écrites dans d'autres langages que celui de 4e Dimension.

La flexibilité et la puissance du langage de 4e Dimension en font l'outil idéal pour toute une gamme de fonctions de gestion de l'information. Les utilisateurs novices peuvent rapidement effectuer des calculs. Les utilisateurs expérimentés peuvent personnaliser leurs bases sans devoir connaître la programmation. Les développeurs plus expérimentés peuvent utiliser la puissance du langage de 4e Dimension pour ajouter à leurs bases de données des fonctions sophistiquées, allant jusqu'au transfert de fichiers et aux communications. Les développeurs maîtrisant la programmation dans d'autres langages peuvent ajouter leurs propres routines au langage de 4e Dimension.

Le langage de programmation de 4e Dimension s'enrichit lorsqu'un des plug-ins de l'environnement 4D est installé dans l'application. Chaque plug-in comporte en effet des commandes spécifiques, permettant de gérer les fonctions qu'il accomplit.

A propos des manuels

Les manuels de 4D, listés ci-dessous, s'appliquent indifféremment à décrire le fonctionnement de 4e Dimension et de 4D Server. Une seule exception : le "Manuel de référence 4D Server", qui traite uniquement des fonctions de 4D Server.

- Le manuel "Langage" est le guide de référence du langage de 4e Dimension. Il vous apprend à personnaliser votre base en utilisant les commandes et fonctions de 4e Dimension.
- Le manuel "Mode Structure" détaille le mode Structure et toutes les opérations que vous pouvez réaliser dans cet environnement.

• Le manuel "Mode Utilisation" décrit le mode Utilisation, l'environnement dans lequel vous exploitez les données de la base et utilisez les formulaires pour traiter les données.
• Le manuel "Prise en main" vous propose de suivre au pas à pas des exemples de création et d'utilisation de bases de données. Ces exercices vous permettent de vous familiariser rapidement avec les fonctionnalités et les concepts de 4e Dimension et 4D Server.
• Le "Manuel de référence 4D Server" est consacré à l'installation et la gestion de bases de données multi-utilisateurs avec 4D Server.

A propos de ce manuel

Ce manuel décrit le langage de 4e Dimension. Nous supposons que vous êtes familiarisé avec le vocabulaire élémentaire utilisé dans 4e Dimension tels que tables, champs ou formulaires. Avant de lire ce manuel, nous vous conseillons de :

• découvrir 4e Dimension à travers les exemples du manuel "Prise en main",
• commencer à créer vos propres bases de données, en vous référant au manuel "Mode Structure",
• vous initier à la gestion des données de votre base dans le mode Utilisation. Reportez-vous pour cela au manuel "Mode Utilisation".

Pour en savoir plus...

Si vous lisez ce manuel pour la première fois, reportez-vous à la section Introduction.

Cette section présente le langage de 4e Dimension. Elle apporte des réponses aux questions suivantes :

- Qu’est-ce que le langage de 4D, et que peut-il vous apporter ?
- Comment utiliser les méthodes ?
- Comment développer une application avec 4e Dimension ?

Ces sujets sont évoqués d'un point de vue général ; chacun d'entre eux est traité plus en détail dans les autres sections de ce manuel.

Qu’est-ce qu’un langage ?

Le langage de 4e Dimension n’est pas très différent de celui que nous utilisons tous les jours. C’est une forme de communication servant à exprimer des idées, informer ou donner des instructions. Tout comme un langage parlé, celui de 4e Dimension se compose d'un vocabulaire, d’une grammaire et d’une syntaxe, que vous employez pour indiquer au programme comment gérer votre base et vos données.

Il n’est pas nécessaire de connaître entièrement le langage. Pour pouvoir vous exprimer en anglais, vous n’êtes pas obligé de connaître la totalité de la langue anglaise ; en réalité, un peu de vocabulaire suffit. Il en va de même pour 4e Dimension — il vous suffit de connaître un minimum du langage pour être productif, vous en apprendrez de plus en plus au fur et à mesure de vos besoins.

Pourquoi utiliser un langage ?

A première vue, un langage de programmation dans 4e Dimension peut sembler inutile. En effet, les modes Structure et Utilisation proposent des outils puissants et entièrement paramétrables, permettant d’effectuer une grande variété de tâches de gestion des données. Les opérations fondamentales telles que la saisie de données, les recherches, les tris et la création d’états sont effectuées facilement. Aussi, de nombreuses autres possibilités sont disponibles, telles que les filtres de validation des données, les aides à la saisie, la représentation graphique et la génération d’étiquettes.

Alors, pourquoi avons-nous besoin d'un langage ? Voici quelques-uns de ses principaux rôles :

• Automatiser les tâches répétitives : par exemple la modification de données, la génération d'états complexes et la réalisation de longues séries d'opérations ne nécessitant pas l'intervention d'un utilisateur.
• Contrôler l'interface utilisateur : vous pouvez gérer les fenêtres, les menus, contrôler les formulaires et les objets d'interface.
• Gérer les données de manière très fine : cette gestion inclut le traitement de transactions, les systèmes de validation complexes, la gestion multi-utilisateurs, l'utilisation des ensembles et des sélections temporaires.
• Contrôler l'ordinateur : vous pouvez contrôler les communications par le port série, la gestion des documents et des erreurs.
• Créer des applications : vous pouvez créer des bases de données personnalisées, faciles à utiliser, en mode Menus créés.
• Ajouter des fonctionnalités aux services Web 4D intégrés : par exemple, vous pouvez créer des pages HTML dynamiques et les insérer parmi celles qui sont automatiquement créées par 4D lorsque les formulaires sont convertis.

Le langage vous permet de contrôler totalement la structure et le fonctionnement de votre base de données. Alors que le mode Utilisation vous offre de puissants outils "génériques", le langage vous offre la possibilité de personnaliser autant que vous voulez votre base de donnée.

Prendre le contrôle de vos données

Le langage de 4e Dimension vous permet de prendre le contrôle total de vos données, d'une manière à la fois puissante et élégante. Il reste d'un abord suffisamment facile pour un débutant, et est suffisamment riche pour un développeur d'applications. Il permet de passer par étapes d'une simple exploitation des fonctions intégrées de 4e Dimension à une base entièrement personnalisée.

Les commandes du langage de 4e Dimension vous laissent la possibilité d'accéder aux éditeurs du mode Utilisation qui vous sont familiers. Par exemple, lorsque vous utilisez la commande CHERCHER, vous accédez à l'éditeur de recherches. Il est presque aussi simple d'utiliser cette commande du langage que de sélectionner la commande **Recherche...** dans le menu **Sélection**, et cela peut même être plus pratique. Vous pouvez, si vous le voulez, indiquer explicitement à la commande CHERCHER ce qu'elle doit rechercher. Par exemple, CHERCHER([Personnes]Nom="Dupont") trouvera toutes les personnes nommées Dupont dans votre base.

Le langage de 4e Dimension est très puissant — une commande remplace souvent des centaines, voire des milliers de lignes de code écrites dans les langages informatiques traditionnels. Paradoxalement, cette puissance s'accompagne d'une réelle simplicité. Les commandes sont écrites en français. Par exemple, vous utilisez la commande CHERCHER pour effectuer une recherche ; pour ajouter un nouvel enregistrement, vous appelez la commande AJOUTER ENREGISTREMENT.

Le langage est organisé de telle manière que vous puissiez facilement accomplir n'importe quelle tâche. L'ajout d'un enregistrement, le tri d'enregistrements, la recherche de valeurs et les autres opérations de même type sont définies par des commandes simples et directes. Mais les routines peuvent également contrôler les ports série, lire des documents sur le disque, traiter des systèmes de transactions complexes, et plus encore.

Même les opérations les plus compliquées se définissent de manière relativement simple. Il serait inimaginable de les réaliser sans le langage de 4D. Cependant, même à l'aide de la puissance des commandes du langage, certaines tâches peuvent se révéler complexes et difficiles. Ce n'est pas l'outil lui-même qui fait le travail ; une tâche peut représenter en soi une difficulté, l'outil ne fait que faciliter son traitement. Par exemple, un logiciel de traitement de texte permet d'écrire un livre plus rapidement et plus facilement, mais il ne l'écrira pas à votre place. Le langage de 4e Dimension vous permet de gérer plus facilement vos données et d'appréhender les tâches ardues en toute confiance.

Est-ce un langage informatique "traditionnel" ?

Si vous êtes familier avec les langages informatiques traditionnels, ce paragraphe peut vous intéresser. Si ce n'est pas le cas, vous pouvez passer directement au paragraphe suivant.

Le langage de 4e Dimension n'est pas un langage informatique traditionnel. C'est un des langages les plus souples et les plus innovants que l'on puisse trouver aujourd'hui sur micro-ordinateur. Le langage a été conçu pour s'adapter à vous, et non l'inverse.

Pour utiliser les langages traditionnels, vous devez avant tout réaliser des maquettes détaillées. C'est même l'une des principales phases d'un développement. 4e Dimension vous permet de commencer à utiliser le langage à tout moment et dans n'importe quelle partie de votre base. Vous pouvez commencer par ajouter une méthode objet dans un formulaire, puis plus tard une ou deux méthodes formulaires. Comme votre base devient plus sophistiquée, vous pouvez ajouter une méthode projet contrôlée par un menu. Vous êtes totalement libre d'utiliser une petite partie du langage ou une partie plus étendue. Ce n'est pas "tout ou rien", comme c'est le cas dans beaucoup d'autres bases de données.

Les langages traditionnels vous obligent à définir et pré-déclarer vos objets sous une forme syntaxique rigide. Dans 4e Dimension, vous créez simplement un objet, comme un bouton, et vous l'utilisez. 4e Dimension gère automatiquement l'objet pour vous. Par exemple, pour utiliser un bouton, il suffit de le dessiner dans un formulaire et de lui donner un nom. Lorsque l'utilisateur clique sur le bouton, le langage déclenche automatiquement vos méthodes.

Les langages traditionnels sont souvent rigides et inflexibles, et exigent que les commandes soient saisies dans un style très formel et contraignant. Le langage de 4e Dimension rompt avec la tradition, au grand bénéfice de l'utilisateur.

Les méthodes, passerelles vers le langage

Une méthode est une série d'instructions qui provoquent l'accomplissement d'une tâche par 4e Dimension. Toute méthode contient une ou plusieurs lignes d'instructions. Chaque ligne d'instruction est composée d'éléments du langage.

Puisque vous avez lu le manuel "Prise en main" de 4e Dimension (vous l'avez lu, n'est-ce pas ?), vous avez déjà écrit et utilisé des méthodes objet et des méthodes projet.

Dans 4e Dimension, vous pouvez créer cinq types de méthodes : des méthodes objet, des méthodes formulaire, des méthodes table ou "triggers", des méthodes projet et des méthodes base.

- **Méthode objet :** méthode généralement courte utilisée pour contrôler un objet de formulaire.
- **Méthode formulaire :** méthode qui gère l'affichage ou l'impression d'un formulaire.
- **Méthode table/trigger :** méthode utilisée pour appliquer les règles de fonctionnement de votre base.
- **Méthode projet :** méthode s'appliquant à la totalité de la base. Les méthodes projet peuvent être associées à des commandes de menus, par exemple.
- **Méthode base :** méthode utilisée pour initialiser des valeurs ou effectuer des actions particulières à l'ouverture ou à la fermeture d'une base, ou lorsqu'un browser Web se connecte à une base publiée comme serveur Web sur un réseau Intranet ou sur Internet.

Les paragraphes suivants présentent chaque type de méthode et expliquent brièvement la manière dont vous pouvez les utiliser pour automatiser votre base.

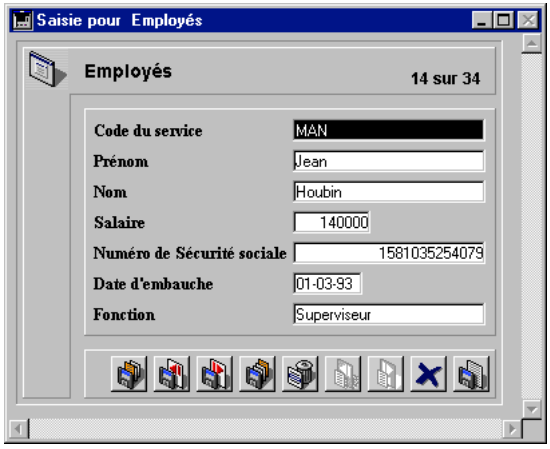
Démarrer avec les méthodes objet

Tout objet d'un formulaire pouvant déclencher une action — c'est-à-dire tout objet actif — peut être associé à une méthode objet. Une méthode objet surveille et gère l'objet actif lors de la saisie et de l'impression des données. Une méthode objet reste liée à un objet actif même lorsque l'objet est copié et collé. Ce principe vous permet de créer des librairies d'objets actifs réutilisables. La méthode objet s'exécute lorsque c'est nécessaire.

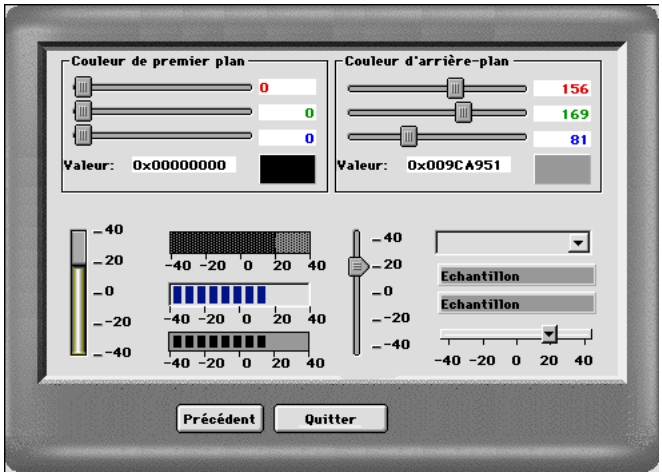
Les méthodes objet sont les outils de base pour la gestion de l'interface utilisateur. L'interface utilisateur est l'ensemble des moyens et des conventions par lesquels l'ordinateur communique avec l'utilisateur. L'interface utilisateur est la porte par laquelle vous entrez dans votre base de données. L'objectif est de la rendre aussi simple d'emploi que possible. L'interface utilisateur doit faire en sorte que l'interaction avec l'ordinateur soit agréable, que l'utilisateur l'apprécie ou, mieux, ne la remarque même pas.

Il y a deux principaux types d'objets actifs dans un formulaire : les objets actifs de saisie, d'affichage et de stockage des données, tels que les champs et les sous-champs, et les objets actifs de contrôle comme les zones de saisie, les boutons, les listes déroulantes et les thermomètres.

4e Dimension vous permet de contruire des formulaires sobres et classiques, tel que celui présenté ci-dessous :



4e Dimension vous permet également de construire des formulaires comportant de multiples éléments de contrôle, comme celui-ci :



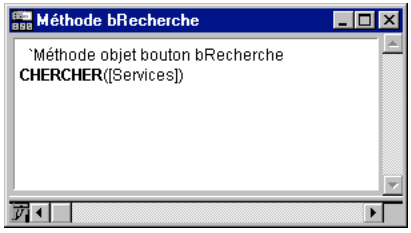
Vous pouvez aussi créer des formulaires originaux en laissant libre cours à votre imagination et en utilisant des éléments graphiques exubérants. Vous n'êtes limité que par votre imagination ou par les souhaits des commanditaires de l'application :

Lorsque vous construisez des formulaires, rappelez-vous que tous les objets actifs disposent de systèmes de contrôle intégrés, comme les intervalles de valeurs et les filtres pour les zones de saisie, ou les actions automatiques pour les objets, menus et boutons.

Essayez toujours d'utiliser ces systèmes intégrés avant de recourir aux méthodes objet. Ils sont comparables aux méthodes objet dans la mesure où ils restent associés aux objets actifs et ne sont exécutés que lorsque l'objet est sollicité. En général, vous utiliserez une combinaison d'aides intégrées et de méthodes objet pour contrôler l'interface utilisateur.

Typiquement, une méthode associée à un objet actif de saisie aura un rôle de gestion des données spécifiquement lié au champ ou à la variable. La méthode peut effectuer un contrôle ou un formatage des valeurs, ou des calculs. Elle peut également récupérer des informations en provenance d'autres tables. Bien entendu, certaines de ces tâches peuvent aussi être exécutées par l'intermédiaire des systèmes intégrés de contrôle de saisie des objets. N'utilisez les méthodes objet que lorsque l'opération à effectuer est plus complexe que ce que permettent ces systèmes. Pour plus d'informations sur les systèmes intégrés de contrôle de saisie, reportez-vous au manuel "Mode Structure" de 4e Dimension.

Des méthodes objet peuvent aussi être associées aux objets de contrôle actifs tels que les boutons. Ces objets sont essentiels pour la navigation au sein de votre base. Les boutons vous permettent de vous déplacer d'un enregistrement à l'autre, de changer de formulaire, d'ajouter et d'effacer des données. Ces objets actifs simplifient l'utilisation d'une base et réduisent le temps d'apprentissage. Les boutons disposent également de systèmes d'aides intégrés et, comme pour la saisie de données, il est préférable de les utiliser avant d'écrire des méthodes objet. Les méthodes vous permettent d'associer à vos objets des actions qui n'existent pas en standard. Dans l'exemple ci-dessous, la méthode objet du bouton affichera l'éditeur de recherches lorsque l'utilisateur cliquera dessus.



A mesure que vous vous familiariserez avec les méthodes objet, vous pourrez créer des librairies d'objets avec leurs méthodes associées. Vous pourrez copier et coller ces objets entre différents formulaires, tables et bases. Vous pourrez même les conserver dans l'Album, prêts à être utilisés suivant vos besoins.

Contrôler des formulaires à l'aide des méthodes formulaire

Tout comme une méthode objet est associée à un objet actif d'un formulaire, une méthode formulaire est associée à un formulaire. Chaque formulaire peut comporter au plus une méthode formulaire.

Un formulaire est un modèle dans lequel vous pouvez visualiser vos données. Les formulaires vous permettent de présenter les données à l'utilisateur de différentes manières. Grâce aux formulaires, vous pouvez créer des écrans de saisie et des états attractifs et faciles à utiliser. Une méthode formulaire contrôle et gère l'utilisation d'un formulaire spécifique, à la fois pour la saisie et l'impression des données.

Une méthode formulaire gère un formulaire à un plus haut niveau que les méthodes objet. Une méthode objet n'est exécutée que lorsque l'objet est utilisé, alors qu'une méthode formulaire est exécutée lorsque n'importe quel objet du formulaire est activé. Les méthodes formulaire sont généralement utilisés pour contrôler l'interaction entre les différents objets et le formulaire dans son ensemble.

Comme les formulaires peuvent être utilisés de nombreuses manières, il est utile de contrôler ce qui se passe lorsque votre formulaire est en cours d'utilisation. Pour cela, 4e Dimension met à votre disposition un grand nombre **d'événements formulaires**. Ces événements vous indiquent précisément ce qui se produit dans le formulaire. Chaque type d'événement (par exemple un clic, un double-clic, une touche du clavier enfoncée...) active ou désactive l'exécution de la méthode formulaire ainsi que les méthodes des objets du formulaire.

Pour plus d'informations sur les formulaires, les objets, les événements et les méthodes, reportez-vous à la description de la commande Evenement formulaire.

Réguler le fonctionnement de votre base à l'aide des méthodes table/triggers

Un trigger est une méthode associée à une table, c'est la raison pour laquelle on peut également parler de méthode table. Les triggers sont automatiquement appelés par le moteur de base de données de 4D à chaque fois qu'une action (ajout, suppression, modification et chargement) est effectuée sur les enregistrements d'une table. Les triggers peuvent empêcher que des opérations interdites soient exécutées avec les enregistrements de votre base. Par exemple, dans un système de facturation, vous ne voulez pas qu'un utilisateur puisse créer des factures sans avoir spécifié le client à qui elle est destinée. Les triggers sont des outils puissants de contrôle des opérations possibles avec les tables, ainsi que de prévention des risques de pertes accidentelles de données. Vous pouvez écrire des triggers très simples, puis les rendre de plus en plus sophistiqués.

Pour plus d'informations sur les triggers, reportez-vous à la section Présentation des triggers.

Utiliser des méthodes projet dans toute la base

A la différence des triggers, des méthodes formulaire et des méthodes objet, qui sont associées à des tables, des formulaires ou des objets actifs particuliers, les méthodes projet sont accessibles depuis n'importe quelle partie de votre base. Les méthodes projet sont réutilisables et peuvent être appelées par d'autres méthodes. Vous pouvez ainsi effectuer plusieurs fois une opération similaire sans devoir écrire plusieurs méthodes.
Vous pouvez appeler des méthodes projet depuis n'importe quelle partie de la base — autres méthodes projet, méthodes objet, méthodes formulaire... Lorsque vous appelez une méthode projet, elle s'exécute comme si elle avait été écrite à l'endroit d'où elle a été appelée. Les méthodes projet appelées depuis d'autres procédures sont appelées sous-routines.
Il y a un autre moyen d'utiliser les méthodes projet : les associer à des commandes de menu. Lorsque vous associez une méthode projet à une commande de menu, la méthode est exécutée lorsque la commande est sélectionnée par l'utilisateur. Vous pouvez considérer la commande de menu comme un appel à une méthode projet.

Gérer les sessions de travail avec les méthodes base

De la même manière que les méthodes objet et formulaire sont exécutées lorsque des événements se produisent dans un formulaire, il existe des méthodes associées à la base qui sont exécutées lorsqu'un événement de type "session de travail" se produit : ce sont les **méthodes base**. Par exemple, à chaque fois que vous ouvrez une base de données, vous voulez initialiser certaines variables qui seront utilisées pendant toute la session de travail ; pour cela, vous pouvez initialiser vos variables dans la méthode base Sur ouverture, qui est automatiquement exécutée par 4D lorsque vous ouvrez la base.

Pour plus d'informations sur les méthode base, reportez-vous à la section Présentation des méthodes base.

Développer votre base de données

La phase de développement est celle pendant laquelle vous personnalisez votre base à l'aide du langage et des autres fonctions intégrées.

En créant simplement une base, vous avez déjà franchi les premières étapes de l'utilisation du langage. Chaque élément d'une base — les tables et les champs, les formulaires et leurs objets — est automatiquement relié au langage. Le langage de 4e Dimension les "reconnaît" tous.

Votre première utilisation du langage pourrait être de créer une méthode objet ou formulaire pour contrôler la saisie de données. Par la suite, vous pourrez définir une méthode formulaire pour gérer l'affichage. Lorsque la base deviendra plus complexe, vous ajouterez une barre de menus avec des méthodes projet pour la personnaliser entièrement.

Comme tous les autres aspects de 4e Dimension, le développement est une phase très souple. Il n'y a aucun cheminement précis à suivre — vous pouvez développer de la manière qui vous semble la plus pratique. Bien entendu, cette phase comprend des étapes générales.

• L'implémentation : définition de la structure et des fonctionnalités de la base en mode Structure.
• Le test : réalisation d'essais de la base en mode Utilisation et de test de chaque élément personnalisé.
• L'utilisation : lorsque la base est entièrement personnalisée, utilisation en mode Menus créés.
• Les corrections : si des erreurs sont détectées, retour au mode Structure pour les corriger.

Des outils particuliers d'aide au développement, n'apparaissant que lorsque c'est nécessaire, sont intégrés à 4e Dimension. A mesure que vous utiliserez le langage de manière de plus en plus intensive, vous vous apercevrez que ces outils facilitent grandement le développement. Par exemple, l'éditeur de méthodes traite automatiquement les éventuelles erreurs de saisie et formate votre travail à la volée ; l'interpréteur (le moteur qui exécute le langage) intercepte les erreurs de syntaxe et vous les désigne ; enfin, le débogueur vous permet de contrôler très précisément l'exécution de vos méthodes afin de détecter toute erreur de structure.

Construire des applications

Vous connaissez maintenant les opérations que l'on peut réaliser avec une base de données — saisie, recherches, tris et créations d'états... Vous avez effectué ces actions dans le mode Utilisation, à l'aide des menus et des éditeurs intégrés.

Lorsque vous utilisez une base de données, vous répétez certaines séquences de tâches. Par exemple, dans une base de contacts personnels, vous pouvez rechercher des contacts, les trier par nom, puis imprimer un état à chaque fois que des informations ont été modifiées. Ces opérations ne sont pas difficiles à réaliser, mais elles peuvent prendre beaucoup de temps lorsqu'il faut les faire 20 fois. De plus, si vous n'avez pas utilisé la base pendant deux semaines, il se peut que vous ayez oublié certaines étapes nécessaires à la création d'un état.

Dans les méthodes, les étapes s'enchaînent les unes après les autres. Ainsi, une seule commande effectue automatiquement toutes les tâches qui lui sont liées. Par conséquent, vous n'avez pas à vous préoccuper des opérations particulières à réaliser.

Dans vos applications, les menus créés permettent d'effectuer des tâches répondant précisément aux besoins de la personne qui utilise la base. Une application se compose de tous les éléments de votre base : la structure, les formulaires, les différentes méthodes, les menus et les mots de passe.

Vous pouvez utiliser 4D Compiler pour compiler vos bases et créer des applications Windows et MacOS autonomes. La compilation des bases de données accélère l'exécution du langage, protège les bases, et vous permet de créer des applications totalement indépendantes. 4D Compiler vérifie également la syntaxe ainsi que les types des variables dans les méthodes, et contrôle donc la cohérence de base.

Les applications que vous créez peuvent aller du plus simple (un menu unique permettant de saisir des noms et d'imprimer un état) au plus complexe (un système de facturation ou de gestion des stocks). Les types d'utilisation des applications sont illimités. Généralement, une application grandit progressivement depuis une base de données en mode Utilisation jusqu'à un logiciel entièrement contrôlé par des menus créés.

Pour en savoir plus...

• Le développement d'applications peut être aussi simple ou aussi complexe que vous le voulez. Pour plus d'informations sur le processus de construction d'une application, reportez-vous à la section Construire une application 4D.
• Si vous débutez avec 4D, reportez-vous aux sections **Présentation du langage** pour découvrir les principes de fonctionnement du langage 4D. Commencez par la section Introduction au langage 4D.

Une application 4D est une base de données conçue pour répondre spécifiquement à des besoins précis. Une application comporte une interface utilisateur destinée à faciliter son utilisation. Toutes les fonctions qu'elles propose sont directement liées (et se limitent) à son champ d'action. 4e Dimension vous permet de créer des applications de manière plus confortable et plus aisée que si vous utilisiez des langages traditionnels.
Parmi la grande variété d'applications que 4e Dimension peut créer, citons :

- un système de facturation,
- un système de gestion des stocks,
- un système de comptabilité,
- un système de paie,
- un gestion des ressources humaines,
- un système de traitement des prospects,
- une base de données accessible par Internet ou Intranet.

Il est parfaitement envisageable qu'une seule application 4D gère tous ces systèmes. Ce type d'applications correspond à une utilisation plutôt traditionnelle des bases de données. De surcroît, les outils proposés par 4e Dimension vous permettent de créer des applications originales, telles que, par exemple :

- un système de gestion documentaire,
- un système graphique de gestion d'images,
- une application de publication de catalogues,
- un système de contrôle et de commande d'un dispositif externe,
- un système de messagerie électronique (E-mail),
- un système multi-utilisateurs de gestion de planning,
- un catalogue recensant les éléments d'une collection de vidéos ou de disques.

Typiquement, une application commence par être une simple base de données exploitée en mode Utilisation. Cette base "se transforme" en application à mesure qu'elle est personnalisée. La caractéristique majeure d'une application est la dissimulation à l'utilisateur des systèmes internes de gestion des fonctions de la base. La gestion de la base est automatisée, et l'utilisateur se sert de menus personnalisés pour exécuter des tâches spécifiques.

Lorsqu'une base 4e Dimension est exploitée en mode Utilisation, vous devez connaître les étapes à atteindre pour obtenir le résultat recherché. Dans une application 4D, c'est le mode Menus créés qui est utilisé. Dans ce mode, vous devez gérer vous-même toutes les fonctions qui sont automatiques dans le mode Utilisation, c'est-à-dire :

• Navigation parmi les tables : les boîtes de dialogue "Sélectionnez la table ou le formulaire" et la Liste des tables ne sont pas accessibles à l'utilisateur. Vous pouvez utiliser les commandes de menus et les méthodes pour contrôler la navigation parmi les tables.
• Menus : en mode Menus créés, seuls les menus Fichier (comportant la commande Quitter), Edition et Aide (ou le menu Pomme sous MacOS) sont affichés par défaut. Si l'application nécessite d'autres menus, vous devez les créer vous-même et les gérer à l'aide des méthodes 4D.
• Editeurs : les éditeurs, tels que les éditeurs de recherches et de tris, ne sont plus automatiquement accessibles en mode Menus créés. Si vous voulez qu'ils restent disponibles, vous devez les appeler en utilisant les commandes du langage.

Les paragraphes suivants fournissent des exemples d'automatisation de l'utilisation d'une base à l'aide du langage.

Menus créés : un exemple

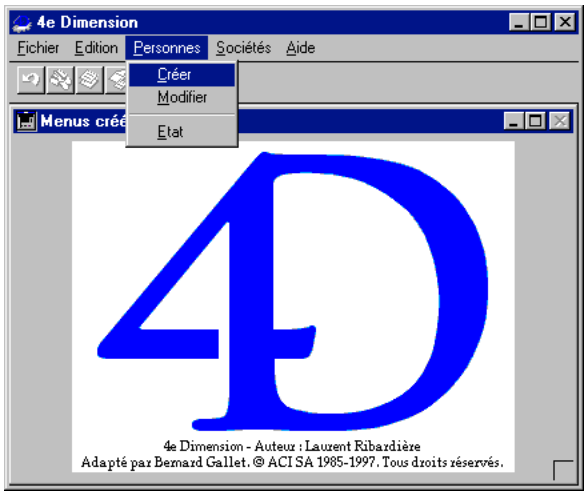
Les menus personnalisés sont les éléments d'interface élémentaires d'une application. La création de menus est très simple — il suffit d'associer une méthode à chaque commande de menu dans l'éditeur de menus.

Les menus personnalisés facilitent l'apprentissage et l'utilisation d'une base de données.

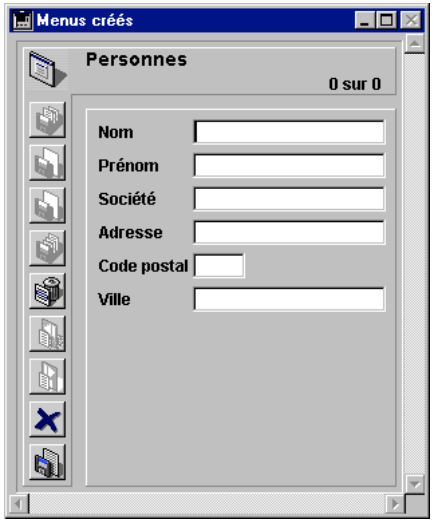
Le paragraphe ci-dessous décrit le point de vue de l'utilisateur lorsqu'il choisit une commande de menu. Le paragraphe suivant détaille ensuite ce qui se produit réellement au cœur l'application ainsi que le travail de conception qui a été effectué. Bien que l'exemple soit simple, il apparaît clairement que la base est plus facile à apprendre et à utiliser. L'utilisateur n'a accès qu'aux éléments correspondant à ses besoins plutôt qu'aux outils "génériques" et aux commandes de menu du mode Utilisation.

Le point de vue de l'utilisateur

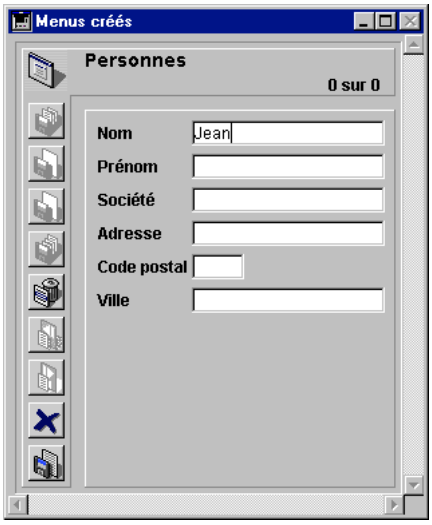
L'utilisation sélectionne une commande de menu personnalisé appelée **Créer** pour ajouter une nouvelle personne dans la base de données.



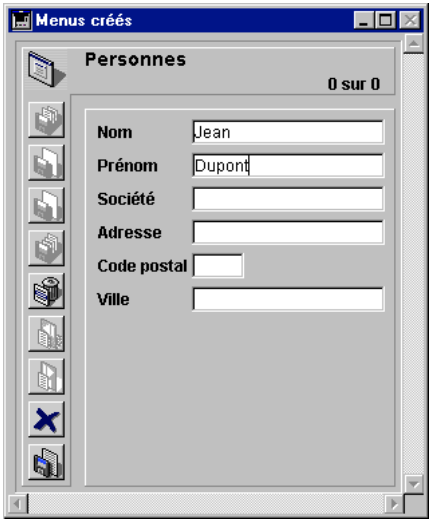
Le formulaire entrée de la table [Personnes] s'affiche.



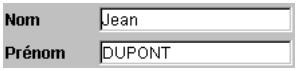
L'utilisateur saisit le prénom de la personne et appuie sur la touche **Tabulation** pour passer au champ suivant.



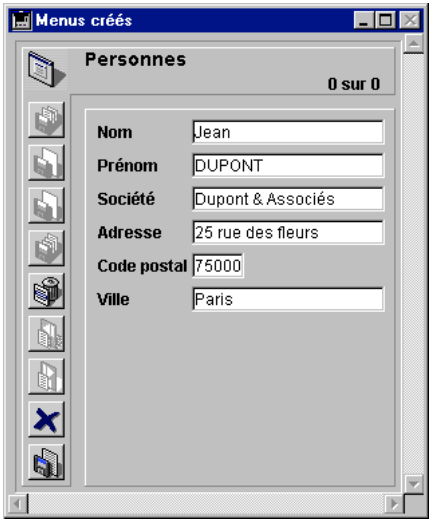
L'utilisateur saisit le nom de la personne et appuie sur la touche **Tabulation** pour passer au champ suivant.



Le nom est converti en lettres majuscules.

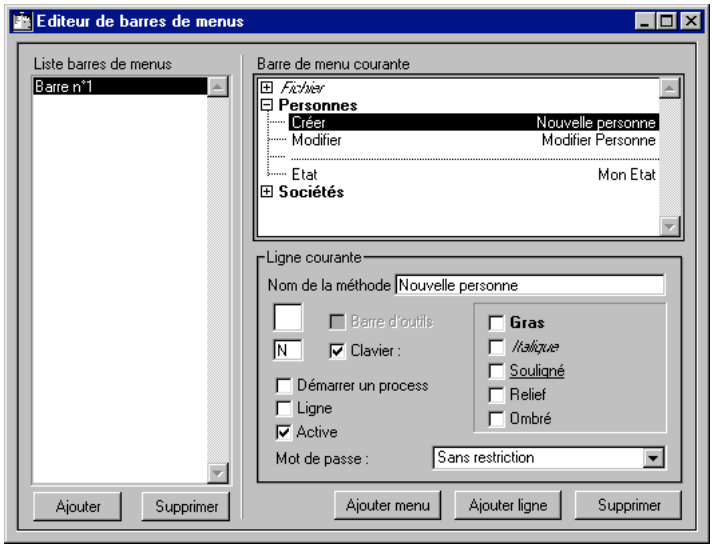


Une fois la saisie terminée, l'utilisateur clique sur le bouton de validation (le dernier dans la colonne de boutons).

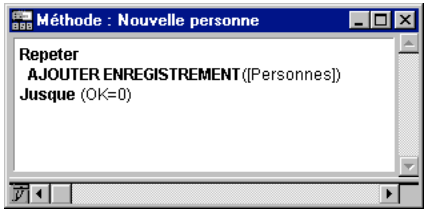


Un autre enregistrement vide s'affiche. L'utilisateur clique sur le bouton **Annuler** (celui qui comporte une croix) pour terminer la "boucle de saisie des données". L'utilisateur retourne à l'écran principal.

Les coulisses
La barre de menus a été créée en mode Structure, à l'aide de l'éditeur de menus.



La commande de menu **Créer** est associée à une méthode projet appelée Nouvelle Personne. Cette méthode a été créée, en mode Structure, dans l'éditeur de méthodes. Lorsque l'utilisateur sélectionne cette commande de menu, la méthode Nouvelle personne s'exécute.



La boucle Repeter...Jusque à l'intérieur de laquelle se trouve la commande AJOUTER ENREGISTREMENT provoque exactement les mêmes effets que la commande de menu **Nouvel enregistrement** en mode Utilisation. Elle affiche le formulaire entrée courant, de manière à ce que l'utilisateur puisse saisir un nouvel enregistrement. Lorsque l'utilisateur valide l'enregistrement, un autre enregistrement vide apparaît. Cette boucle AJOUTER ENREGISTREMENT continue de s'exécuter jusqu'à ce que l'utilisateur clique sur le bouton **Annuler**.

Lorsque l'utilisateur remplit un enregistrement, les actions suivantes sont déclenchées :
• Comme il n'y a pas de méthode objet associée au champ Prénom, rien ne s'exécute.
• Une méthode est associée au champ Nom. Cette méthode a été créée, en mode Structure, à l'aide des éditeurs de formulaires et de méthodes. Elle exécute l'instruction suivante :

```
Nom:=Majusc(Nom)
```

Cette ligne convertit le champ "Nom" en caractères majuscules.

Une fois qu'un enregistrement a été saisi, lorsque l'utilisateur clique sur le bouton d'annulation dans le formulaire suivant, la variable système OK prend la valeur zéro, ce qui constitue la condition d'arrêt de l'exécution de la boucle AJOUTER ENREGISTREMENT.

Comme il n'y a pas d'autres instructions à exécuter, la méthode Nouvelle Personne stoppe son exécution et rend la main à la barre de menus principale.

Comparer une application 4D avec le mode Utilisation

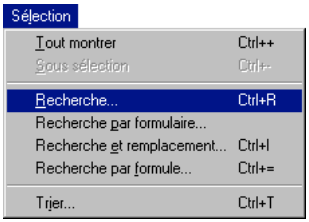
Comparons la manière dont une même tâche est effectuée en mode Utilisation et à l'aide du langage.

Examinons une opération courante :
• Sélectionner un groupe d'enregistrements,
• Le trier,
• Imprimer un état.

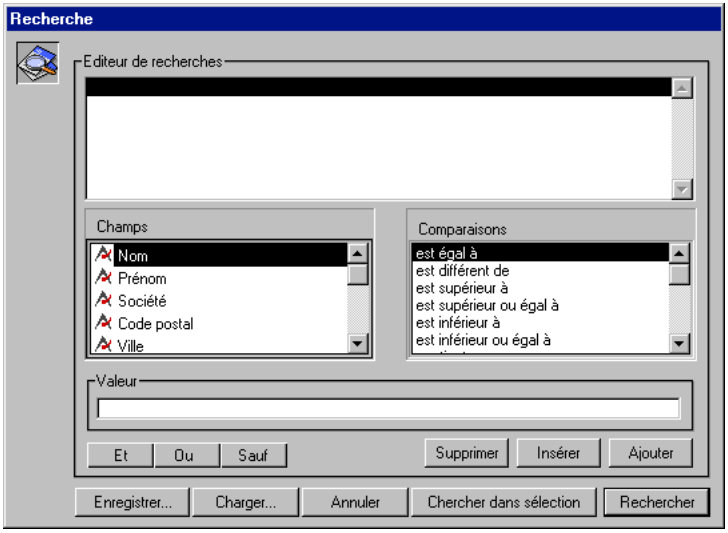
Le paragraphe ci-dessous, "Exploiter une base en mode Utilisation", traite de la réalisation de ces tâches à partir du mode Utilisation. Le paragraphe suivant, "Exploiter une application 4D avec les éditeurs intégrés", traite de la réalisation des mêmes tâches à partir d'une application en mode Menus créés.

Notez que, bien que les mêmes opérations sont effectuées dans les deux cas, les étapes dans le second paragraphe sont automatisées par programmation.

Exploiter une base en mode Utilisation
L'utilisateur choisit la commande **Recherche...** dans le menu **Sélection**.

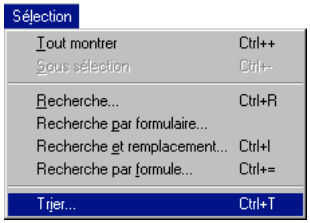


L'éditeur de recherches s'affiche :

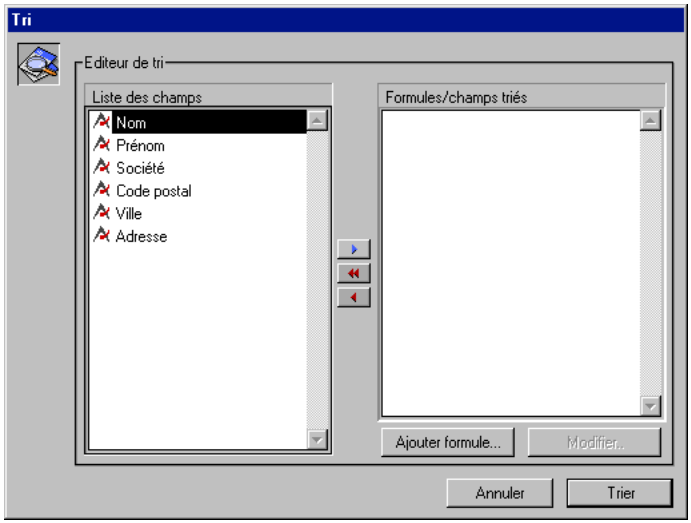


L'utilisateur définit ses critères de recherche et clique sur le bouton OK. La recherche s'effectue.

L’utilisateur choisit la commande **Trier...** dans le menu **Sélection**.



L'éditeur de tris s’affiche.



L’utilisateur définit ses critères de tri et clique sur le bouton **Trier**. Le tri est effectué.

Puis, pour imprimer les enregistrements, les étapes suivantes sont nécessaires :
• L’utilisateur choisit la commande **Imprimer** dans le menu **Fichier**.
La boîte de dialogue de choix du formulaire d’impression s’affiche (l’utilisateur doit savoir quel formulaire choisir !).
• Les boîtes de dialogue de l’imprimante sélectionnée s’affichent. L’utilisateur fixe ses paramètres, puis l’état est imprimé.

Exploiter une application 4D avec les éditeurs intégrés

Examinons maintenant comment ces opérations peuvent être effectuées en mode Menus créés.

L'utilisateur choisit la commande **Etat** dans le menu **Personnes**.

Même à ce stade, l'utilisation d'une application apparaît plus facile. L'utilisateur n'a pas besoin de savoir qu'il faut commencer par effectuer une recherche.

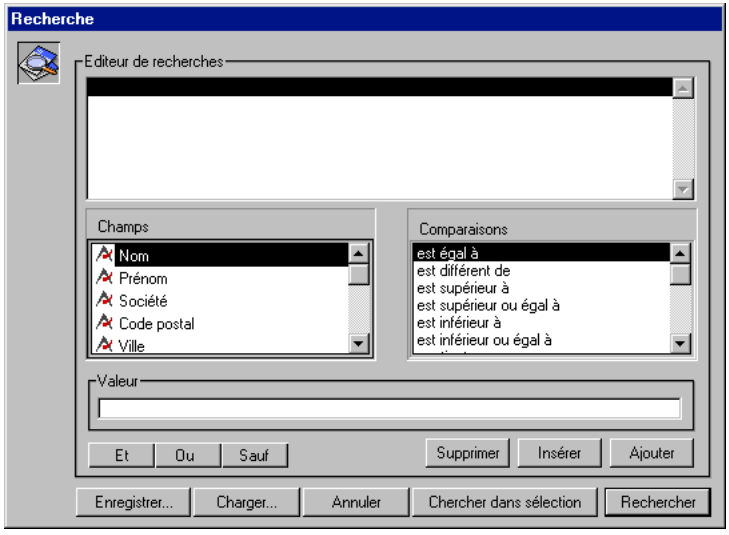
Une méthode appelée Mon Etat est associée à la commande de menu. Elle se présente ainsi :

```
CHERCHER ([Personnes])
TRIER ([Personnes])
FORMULAIRE SORTIE ([Personnes]; "Etat")
IMPRIMER SELECTION ([Personnes])
```

La première ligne est exécutée :

```
CHERCHER ([Personnes])
```

L'éditeur de recherches s'affiche.



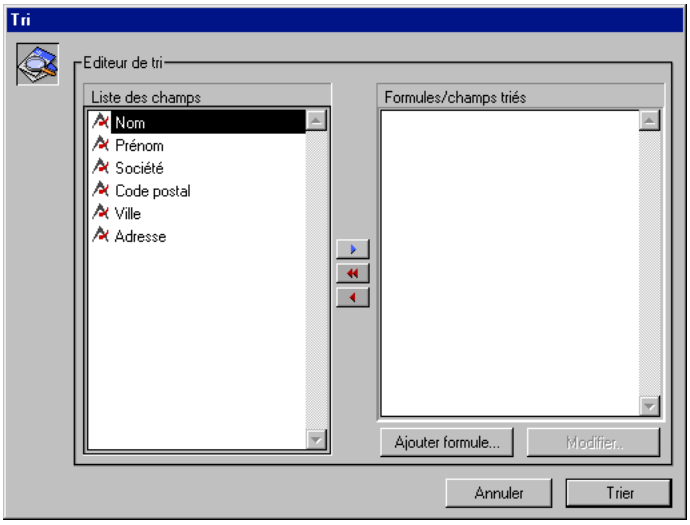
L'utilisateur définit ses critères de recherche et clique sur le bouton **Rechercher**. La recherche s'effectue.

La deuxième ligne de la méthode Mon Etat est exécutée :

TRIER ([Personnes])

Vous notez que l'utilisateur n'avait pas besoin de savoir que le tri était la deuxième étape.

La boîte de dialogue de tri s'affiche.



L'utilisateur définit ses critères de tri et clique sur le bouton **Trier**. Le tri est effectué.

La troisième ligne de Mon Etat est exécutée :

FORMULAIRE SORTIE ([Personnes]; "Etat")

Une fois de plus, il n'est pas nécessaire que l'utilisateur sache ce qu'il faut faire ensuite. Le cheminement est déjà tracé.

La dernière ligne de la méthode Mon Etat est exécutée :

IMPRIMER SELECTION ([Personnes])

Les boîtes de dialogue de l'imprimante sélectionnée s'affichent. L'utilisateur fixe ses paramètres, puis l'état est imprimé.

Automatiser davantage l'application

Les mêmes commandes que celles qui ont été employées dans la comparaison précédente peuvent être utilisées pour automatiser encore plus la base de données.

Examinons la nouvelle version de la méthode Mon Etat.

L'utilisateur choisit **Etat** dans le menu **Personnes**. La méthode appelé Mon Etat2 est associée à la commande de menu :

```
CHERCHER ([Personnes]; [Personnes]Entreprise = "Dupont & Associés")
TRIER ([Personnes]; [Personnes]Nom; > ; [Personnes]Prénom; >)
FORMULAIRE SORTIE([Personnes]; "Etat")
IMPRIMER SELECTION ([Personnes]; *)
```

La première ligne est exécutée :

```
CHERCHER ([Personnes]; [Personnes]Entreprise = "Dupont & Associés")
```

L'éditeur de recherches ne s'affiche pas. Au lieu de cela, la recherche est définie et effectuée par la commande CHERCHER. L'utilisateur n'a rien à faire.

La deuxième ligne est exécutée :

```
TRIER ([Personnes]; [Personnes]Nom; > ; [Personnes]Prénom; >)
```

La boîte de dialogue de tri n'est pas affichée, et le tri est immédiatement effectué. Une fois encore, aucune intervention de l'utilisateur n'est nécessaire.

Les dernières lignes de la méthode Mon Etat2 sont exécutées :

```
FORMULAIRE SORTIE([Personnes]; "Etat")
IMPRIMER SELECTION ([Personnes]; *)
```

Les boîtes de dialogue standard d'impression ne sont pas affichées. La commande IMPRIMER SELECTION comporte en effet le paramètre optionnel astérisque (*), ce qui lui indique d'utiliser les paramètres d'impression en vigueur au moment où le formulaire d'état a été créé. L'état est imprimé.

Les avantages de cette automatisation accrue sont les suivants :
• La recherche est effectuée automatiquement : cela évite que les utilisateurs sélectionnent de mauvais critères lorsqu'ils construisent leur recherche.
• Le tri est effectué automatiquement : cela évite que les utilisateurs sélectionnent de mauvais critères lorsqu'ils construisent leur tri.

• L'impression est effectuée automatiquement : cela évite que les utilisateurs sélectionnent un formulaire inadapté.
• Vous évitez à l'utilisateur d'avoir à définir des options dans les trois boîtes de dialogue.

Outils supplémentaires de développement d’applications 4D

A mesure que vous progresserez dans votre développement d'une application 4D, vous allez découvrir de nombreuses fonctionnalités que vous n'aviez pas vues lorsque vous avez commencé.
Mais vous pouvez aller encore plus loin, en ajoutant des outils et des plug-ins dans votre environnement 4D.

Outils de développement

4D fournit plusieurs outils permettant de développer des applications. Ces outils vous permettent de déplacer des objets d’une base à l’autre, de compiler vos bases, et vous aident à vérifier la syntaxe de votre code. Ces outils sont les suivants :

• **4D Insider** vous permet de construire des tables de référence croisée de vos bases 4e Dimension. Vous pouvez visualiser et imprimer vos méthodes, variables, commandes, externes, tables, listes et formulaires. L’éditeur de références croisées désigne tous les endroits de votre base où vos objets sont utilisés. 4D Insider vous permet aussi de déplacer des objets tels que des tables, formulaires, méthodes, barres de menus, énumérations, externes et styles d’une base à une autre.

• **4D Compiler** traduit vos méthodes directement en langage machine (assembleur), ce qui accélère la vitesse d’exécution de vos bases de données. 4D Compiler teste également la cohérence de votre code et détecte tout conflit logique ou de syntaxe. De plus, ce programme protège votre base et empêche toute modification non autorisée.

Plug-Ins 4D

Vous pouvez augmenter les capacités de vos applications 4D en ajoutant des **Plug-Ins** professionnels à votre environnement de développement 4D. 4D fournit les plug-ins de **productivité** suivants :

• **4D Write** : Traitement de textes
• **4D Draw** : Créateur de documents graphiques de type CAD.

4D fournit également les plug-ins de **connectivité** suivants :

• **4D ODBC** : Connectivité via ODBC
• **4D for ORACLE** : Connectivité avec des bases ORACLE
• **4D Open for java** : Connectivité avec les applications JAVA
• **4D Open for 4D** : Connectivité de 4D à 4D pour construire des systèmes 4D d'information distribuée.

Pour plus d'informations, contactez 4D ou un Partenaire 4D, ou visitez notre site Web :

http://www.4d.fr

La communauté 4D et les produits des partenaires 4D

Il existe une communauté internationale très active organisée autour de 4D, composée de groupes d'utilisateurs, de forums électroniques et de partenaires 4D, qui développe des **produits liés à 4D**.

Examinez le contenu de votre CD 4D, il contient des versions de démonstration et des informations en provenance des Partenaires 4D. Vous pourrez les contacter sur le Web. Vous pouvez souscrire un abonnement à la revue indépendante **Planète 4D** (http://www.planete4D.com). La communauté 4D vous fournira de nombreux conseils et solutions, ainsi que des produits tiers vous permettant d'augmenter votre productivité. Vous aurez accès à une foule d'informations et d'astuces qui vous feront économiser votre temps et votre énergie.

2

Présentation du langage

Le langage de 4e Dimension est constitué de différents composants, vous permettant d'effectuer de multiples opérations et de gérer vos données.

• **Types de données** : Catégories de données stockées dans une base. Ce point est traité dans le paragraphe suivant. Pour une description détaillée, référez-vous à la section Types de données.
• **Variables** : Adresses de stockage temporaires de données en mémoire. Pour une description détaillée, référez-vous à la section Variables.
• **Opérateurs** : Symboles effectuant un calcul entre deux valeurs. Ce point est traité dans les paragraphes suivants. Pour une description détaillée, référez-vous à la section Opérateurs et à ses sous-sections.
• **Expressions** : Combinaisons de composants du langage ayant pour résultat le renvoi d'une valeur. Ce point est traité dans les paragraphes suivants.
• **Commandes** : Instructions intégrées effectuant une opération. Toutes les commandes de 4D (par exemple AJOUTER ENREGISTREMENT) sont décrites dans ce manuel, groupées par thème. Lorsque c'est nécessaire, les thèmes comprennent une section d'introduction. Vous pouvez utiliser des Plug-ins 4D pour ajouter de nouvelles commandes à votre environnement de développement 4D. Par exemple, une fois que vous avez installé le plug-in 4D Write dans votre base de données, vous pouvez utiliser les commandes de 4D Write pour créer et manipuler des fichiers de traitement de texte.
• **Méthodes** : Instructions que vous écrivez à l'aide de tous les éléments du langage décrits ci-dessus. Pour une description détaillée, référez-vous à la section Méthodes et à ses sous-sections.

Cette section présente les Types de données, les Opérateurs et les Expressions. Pour une description d'un autre composant, reportez-vous aux sections précédemment citées.

De plus :
• Les composants du langage tels que les variables sont identifiés par leur nom. Pour une description détaillée des règles de définition des noms d'objets, reportez-vous à la section Nommer les objets du langage 4D.
• Pour plus d'informations sur les variables tableaux, reportez-vous à la section Présentation des tableaux.
• Pour plus d'informations sur les variables BLOB, reportez-vous à la section Commandes du thème BLOB.
• Si vous avez l'intention de compiler votre base, reportez-vous à la section Commandes du thème Compilateur ainsi qu'au manuel de référence de 4D Compiler.

Types de données

De nombreux types de données peuvent être stockés dans une base 4e Dimension. Le langage distingue sept types de données élémentaires : chaîne, numérique, date, heure, booléen, image et pointeur.

• **Chaîne** : Une suite de caractères, telles que “Bonjour à tous”. Les champs et variables alpha et texte sont des données de type Chaîne.
• **Numérique** : Nombres, tels que 2 ou 1 000,67. Les champs et variables entier, entier long et numérique (aussi appelé réel) sont des données de type Numérique.
• **Date** : Dates telles que 20/11/97. Les champs et variables date sont des données de type Date.
• **Heures** : Heures, c’est-à-dire heures, minutes et secondes, telles que 21:00:00 ou 4:35:30 de l'après-midi. Les champs et variables heure sont des données de type Heure.
• **Booléen** : Valeurs logiques de VRAI ou FAUX. Les champs et variables booléens sont des données de type Booléen.
• **Image** : Les champs et variables image sont des données de type Image.
• **Pointeur** : Type spécial de données, utilisé en programmation avancée. Les variables pointeur sont des données de type Pointeur. Il n’y a pas de type de champ correspondant.

Vous constatez que, dans cette liste de types de données, les types chaîne et numérique sont associés à plus d’un type de champ. Lorsque des données sont placées dans un champ, le langage les convertit automatiquement dans le type du champ. Par exemple, si un champ de type entier est utilisé, les valeurs qu’il contient sont automatiquement traitées en tant que numériques. En d’autres termes, vous n’avez pas à vous préoccuper du mélange de champs de types semblables lorsque vous programmez avec 4D ; le langage le gère pour vous.

Cependant, il est important, lorsque vous utilisez le langage, de ne pas mélanger les types de données différents. Tout comme il est absurde de stocker la valeur “ABC” dans un champ de type Date, il est absurde de donner la valeur “ABC” à une variable utilisée pour des dates. Dans la plupart des cas, 4e Dimension est très tolérant et tentera d’utiliser de manière logique ce que vous faites. Par exemple, si vous additionnez un nombre x et une date, 4e Dimension déduira que vous voulez ajouter x jours à la date, mais si vous tentez d’ajouter une chaîne à une date, 4e Dimension vous préviendra que cette opération est impossible.

Certains cas nécessitent que vous stockiez des données dans un type et que vous les utilisiez dans un autre. Le langage contient un ensemble complet de commandes vous permettant de convertir des types de données vers d'autres types. Par exemple, si vous voulez créer un numéro de matricule commençant par des chiffres et se terminant par des lettres, vous pouvez écrire :

```
[Produits]Matricule:=Chaine(Numéro)+"abc"
```

où, si Numéro vaut 17, [Produits]Matricule prendra la valeur "17abc".

Les types de données sont détaillés dans la section Types de données.

Opérateurs

Lorsque vous programmez avec 4D, il est rare que vous ayez simplement besoin de données "brutes". Le plus souvent, il sera nécessaire de traiter ces données d'une manière ou d'une autre. Vous effectuez ces calculs avec des **opérateurs**. Les opérateurs, en général, prennent deux valeurs et effectuent avec elles une opération dont le résultat est une troisième valeur. Vous connaissez déjà la plupart des opérateurs. Par exemple, 1 + 2 utilise l'opérateur d'addition (ou signe plus) pour faire la somme de deux nombres, et le résultat est 3. Le tableau ci-dessous présente quelques opérateurs courants :

Opérateur	Opération	Exemple
+	Addition	1 + 2 ce qui fait 3
–	Soustraction	3 – 2 ce qui fait 1
*	Multiplication	2 * 3 ce qui fait 6
/	Division	6 / 2 ce qui fait 3

Les opérateurs numériques ne représentent qu'un seul des différents types d'opérateurs disponibles. Comme 4e Dimension traite de multiples types de données, tels que des nombres, des dates ou des images, vous disposez d'opérateurs particuliers effectuant des opérations sur ces données.

Souvent, les mêmes symboles sont utilisés pour des opérations différentes, en fonction du type de données traitées. Par exemple, le signe (+) peut effectuer diverses opérations, comme le montre le tableau suivant :

Type de données	Opération	Exemple
Numérique	Addition	1 + 2 ajoute les nombres, le résultat est 3
Chaîne	Concaténation	“Bonjour ” + “à tous” concatène (met bout à bout) les chaînes, le résultat est “Bonjour à tous”
Date et Numérique	Addtion de date	!1/1/1997! + 20 ajoute 20 jours à la date 1 janvier 1997, le résultat est la date 21 janvier 1997

Les opérateurs sont détaillés dans la section Opérateurs et ses sous-sections.

Expressions

Pour parler simplement, les expressions retournent une valeur. En fait, lorsque vous programmez avec 4e Dimension, vous utilisez tout le temps des expressions et vous avez tendance à les manipuler uniquement à travers la valeur qu’elles représentent. Les expressions sont aussi appelées formules.

Les expressions peuvent être constituées de presque tous les composants du langage : commandes, opérateurs, variables et champs. Vous utilisez des expressions pour écrire des lignes de code, qui sont à leur tour utilisées pour construire des méthodes. Des expressions sont employées à chaque fois que le langage de 4D a besoin de connaître la valeur d’une donnée.

Les expressions sont rarement “indépendantes”. Il n’y a que peu d’endroits dans 4e Dimension où une expression peut être utilisée en tant que telle : dans la boîte de dialogue de Recherche par formule en mode Utilisation, dans la fenêtre du Débogueur où la valeur des expressions peut être évaluée, dans la boîte de dialogue Appliquer une formule, et dans l’éditeur d’états semi-automatiques en tant que formule dans une colonne.

Une expression peut être simplement une constante, telle que le chiffre 4 ou la chaîne “Bonjour”. Comme son nom l’indique, la valeur d’une constante ne change jamais. C’est lorsqu’elles contiennent des opérateurs que les expressions commencent à devenir intéressantes. Dans les paragraphes précédents, vous avez déjà pu voir des expressions utilisant des opérateurs. Par exemple, 4 + 2 est une expression qui utilise l’opérateur d’addition pour additionner deux nombres, et dont le résultat est 6.

Vous vous référez à une expression par l'intermédiaire du type de données qu'elle retourne. Il existe sept types d'expressions :

- Expression chaîne,
- Expression numérique (aussi appelée nombre),
- Expression date,
- Expression heure,
- Expression booléenne,
- Expression image,
- Expression pointeur.

Le tableau suivant fournit un exemple pour chacun des sept types d'expressions.

Expression	Type	Description
"Bonjour"	Chaîne	Le mot Bonjour est une constante chaîne, signalée par les guillemets.
"Bonjour " + "à tous"	Chaîne	Deux chaînes, "Bonjour " et "à tous", sont mises bout à bout (concaténées) à l'aide de l'opérateur de concaténation de chaînes (+). La chaîne "Bonjour à tous" est retournée.
"M. " + [Amis]Nom	Chaîne	Deux chaînes sont concaténées : la chaîne "M." et la valeur courante du champ Nom de la table Amis. Si le champ contient "Dupont", l'expression retourne "M. Dupont".
Majusc ("dupont")	Chaîne	Cette expression utilise "Majusc", une commande du langage, pour convertir la chaîne "dupont" en majuscules. Elle retourne "DUPONT".
4	Numérique	C'est une constante numérique, 4.
4 * 2	Numérique	Deux nombres, 4 et 2, sont multipliés à l'aide de l'opérateur de multiplication (*). Le résultat est le nombre 8.
MonBouton	Numérique	C'est le nom d'un bouton. Il retourne la valeur courante du bouton : 1 s'il y a eu un clic sur le bouton, 0 sinon.
!25/1/97!	Date	C'est une constante date pour la date 25/01/97 (25 janvier 1997). Notez l'emploi des points d'exclamation pour indiquer une constante date.

Date du jour + 30	Date	C'est une expression de type Date qui utilise la fonction Date du jour pour récupérer la date courante. Elle ajoute 30 jours à la date d'aujourd'hui et retourne la nouvelle date.
?8:05:30?	Heure	C'est une constante heure qui représente 8 heures, 5 minutes, et 30 secondes.
?2:03:04? + ?1:02:03?	Heure	Cette expression ajoute une heure à une autre et retourne l'heure 3:05:07.
Vrai	Booléen	Cette commande retourne la valeur booléenne VRAI.
10 # 20	Booléen	C'est une comparaison logique entre deux nombres. Le symbole (#) signifie “est différent de”. Comme 10 “est différent de” 20, l'expression retourne VRAI.
“ABC” = “XYZ”	Booléen	C'est une comparaison logique entre deux chaînes. Elles sont différentes, donc l'expression retourne FAUX.
MonImage + 50	Image	Cette expression considère l'image placée dans MonImage, la déplace de 50 pixels vers la droite, et retourne l'image résultante.
->[Amis]Nom	Pointeur	Cette expression retourne un pointeur vers le champ [Amis]Nom.
Table (1)	Pointeur	C'est une commande qui retourne un pointeur vers la première table.

Référence

Constantes, Méthodes, Opérateurs, Pointeurs, Présentation des tableaux, Types de données, Variables.

Les champs, variables et expressions de 4e Dimension ont un type représentant les données qu'ils contiennent. 4e Dimension accepte le typage de ces éléments en fonction du tableau suivant :

Types de données	Champ	Variable	Expression
Chaîne (cf. note 1)	Oui	Oui	Oui
Numérique (cf. note 2)	Oui	Oui	Oui
Date	Oui	Oui	Oui
Heure	Oui	Oui	Oui
Booléen	Oui	Oui	Oui
Image	Oui	Oui	Oui
Pointeur	Non	Oui	Oui
BLOB (cf. note 3)	Oui	Oui	Non
Tableau (cf. note 4)	Non	Oui	Non
Sous-table	Oui	Non	Non
Indéfini	Non	Oui	Oui

Notes
(1) Une chaîne peut être un champ alphanumérique, une variable de longueur fixe, ou encore une variable ou un champ de type Texte.
(2) Un numérique peut être une variable ou un champ de type Réel (Numérique), Entier et Entier long.
(3) BLOB est l'abréviation de Binary Large OBject. Pour plus d'informations sur les BLOBs, reportez-vous à la section Commandes du thème BLOB.
(4) Les tableaux peuvent être de tout type. Pour plus d'informations, reportez-vous à la section Présentation des tableaux.

Chaîne

Chaîne est un terme générique utilisé pour :
• les champs alphanumériques
• les variables de longueur fixe
• les variables ou champs de type Texte
• toute expression de type Alpha ou Texte

Une chaîne se compose de caractères. Chaque caractère peut être l'un des 256 caractères ASCII supportés par Windows et MacOS. Pour plus d'informations sur les codes ASCII, reportez-vous à la section Codes ASCII.

• Un champ alphanumérique peut contenir de 0 à 80 caractères (la limite est fixée lors de la définition du champ).
• Une variable de longueur fixe peut contenir de 0 à 255 caractères (la limite est fixée lors de la déclaration de la variable).
• Un champ, une variable ou une expression de type Texte peut contenir de 0 à 32 000 caractères.

Vous pouvez assigner un alpha à un texte et vice-versa, 4D effectue automatiquement la conversion, en tronquant les valeurs si nécessaire. Vous pouvez mélanger du texte et de l'alphanumérique dans les expressions.

Note : Dans ce manuel de référence du langage 4D, les paramètres de type Alpha et Texte dans les descriptions des commandes sont indifféremment appelés Chaîne, sauf spécification explicite.

Numérique

Numérique est un terme générique utilisé pour :
• les champs, variables ou expression de type Réel (aussi appelé type Numérique)
• les champs, variables ou expression de type Entier
• les champs, variables ou expression de type Entier long

Les nombres de type Réel sont compris dans l'intervalle ±1.7e±308 (15 chiffres).
Les nombres de type Entier (2 octets) sont compris dans l'intervalle -32 768..32 767
Les nombres de type Entier long (4 octets) sont compris dans l'intervalle -2^31..(2^31)-1

Vous pouvez assigner tout nombre d'un type numérique à un nombre d'un autre type numérique, 4D effectue automatiquement la conversion, en tronquant ou en arrondissant les valeurs si nécessaire. Notez cependant que lorsqu'une valeur est située en-dehors de l'intervalle du type de destination, 4D ne pourra la convertir. Vous pouvez mélanger tous les types de numériques au sein d'une même expression.

Note : Dans ce manuel de référence du langage 4D, quel que soit le type précis des données, les paramètres de type Réel, Entier et Entier long dans les descriptions des commandes sont appelés Numériques, sauf spécification explicite.

Date

• Les variables, champs ou expressions de type Date peuvent être compris entre 1/1/100 et 31/12/32767.
• Une date est structurée sous la forme jour/mois/année (sur un système français).
• Si l'année est fournie avec deux chiffres seulement, 4e Dimension suppose que le siècle est le 20e si la valeur est supérieure ou égale à 30 et le 21e si elle est inférieure (ce comportement par défaut peut être modifié à l'aide de la commande SIECLE PAR DEFAUT).

Note : Dans ce manuel de référence du langage 4D, les paramètres de type Date dans les descriptions des commandes sont appelés Date, sauf spécification explicite.

Heure

• Les variables, champs ou expressions de type Heure peuvent être compris entre 00:00:00 et 596 000:00:00.
• Une heure est structurée sous la forme heure:minute:seconde (avec une version française de 4D).
• Les heures sont stockées dans un format de 24 heures.
• Une valeur de type Heure peut être traitée en tant que nombre. Le nombre correspondant est le nombre de secondes que cette valeur représente. Pour plus d'informations, reportez-vous à la section Opérateurs sur les heures.

Note : Dans ce manuel de référence du langage 4D, les paramètres de type Heure dans les descriptions des commandes sont appelés Heure, sauf spécification explicite.

Booléen

Les variables, champs ou expressions de type Booléen peuvent être soit à VRAI soit à FAUX.

Note : Dans ce manuel de référence du langage 4D, les paramètres de type Booléen dans les descriptions des commandes sont appelés Booléen, sauf spécification explicite.

Image

Les variables, champs ou expressions de type Image peuvent contenir des images Windows ou Macintosh. En général, ce type accepte toute image pouvant être collée dans le Presse-papiers ou bien lue depuis le disque à l'aide des commandes de 4D ou d'un plug-in.

Note : Dans ce manuel de référence du langage 4D, les paramètres de type Image dans les descriptions des commandes sont appelés Image, sauf spécification explicite.

Pointeur

Les variables ou expressions de type Pointeur sont des références à d'autres variables (y compris des tableaux et des éléments de tableaux), à des tables ou à des champs. Il n'existe pas de champs de type Pointeur.

Pour plus d'informations sur les pointeurs, reportez-vous à la section Pointeurs.

Note : Dans ce manuel de référence du langage 4D, les paramètres de type Pointeur dans les descriptions des commandes sont appelés Pointeur, sauf spécification explicite.

BLOB

Les champs ou variables de type BLOB sont des séries d'octets (d'une longueur de 0 à 2 Go) que vous pouvez adresser individuellement ou à l'aide des Commandes du thème BLOB. Il n'existe pas d'expressions de type BLOB.

Note : Dans ce manuel de référence du langage 4D, les paramètres de type BLOB dans les descriptions des commandes sont appelés BLOB.

Tableau

Les tableaux ne sont pas véritablement un type de données. Sous cette appellation sont regroupés les différents types de tableaux (comme les tableaux entier, tableaux texte, etc.). Les tableaux sont des variables. Il n'existe pas de champs ni d'expressions de type Tableau. Pour plus d'informations sur les tableaux, reportez-vous à la section Présentation des tableaux.

Note : Dans ce manuel de référence du langage 4D, les paramètres de type Tableau dans les descriptions des commandes sont appelés Tableau, sauf spécification explicite (par exemple Tableau Alpha, Tableau Réel...).

Sous-table

Les sous-tables ne sont pas véritablement un type de données. Seuls les champs peuvent être de type Sous-table. Il n'existe pas de variables ni d'expressions de type Sous-table. Pour plus d'informations sur les sous-tables, reportez-vous au manuel *Mode Structure* ainsi qu'aux commandes du thème Sous-enregistrements.

Indéfini

Indéfini n'est pas véritablement un type de données. Une variable dite "indéfinie" est une variable n'ayant pas encore été définie. Une fonction utilisateur (c'est-à-dire une méthode projet qui retourne une valeur) peut retourner une valeur indéfinie si, à l'intérieur de la méthode, le résultat de la fonction ($0) est assigné à une expression indéfinie (une expression issue d'un calcul effectué avec au moins une variable indéfinie). Un champ ne peut pas être indéfini.

Convertir les types de données

Le langage de 4D comporte des fonctions et des opérateurs vous permettant de convertir des types de données en d'autres types, dans la mesure où de telles conversions ont un sens. 4D assure la vérification des types de données. Ainsi, vous ne pouvez pas écrire : "abc"+0.5+!25/12/96!-?00:30:45?, car cette opération génère une erreur de syntaxe.

Le tableau ci-dessous liste les types de données pouvant être convertis, le type dans lequel ils peuvent être convertis, ainsi que les fonctions 4D à utiliser.

Types à convertir	en Chaîne	en Numérique	en Date	en Heure
Chaîne		**Num**	**Date**	**Heure**
Numérique (*)	**Chaine**			
Date	**Chaine**			
Heure	**Chaine**			
Booléen		**Num**		

(*) Les valeurs de type Heure peuvent être utilisées en tant que numériques.

Note : Ce tableau ne traite pas les conversions de données plus complexes obtenues à l'aide d'une combinaison d'opérateurs et d'autres commandes.

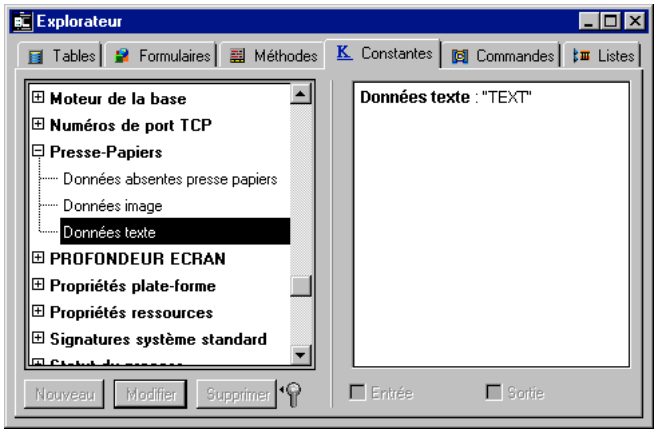
Référence

Conditions et boucles, Constantes, Méthodes, Nommer les objets du langage 4D, Opérateurs, Pointeurs, Présentation des tableaux, Type, Variables.

Une constante est une expression dont la valeur est fixe. Il existe deux types de constantes : les **constantes prédéfinies** que vous pouvez appeler en inscrivant leur nom et les **constantes littérales**, pour lesquelles vous devez saisir une valeur.

Constantes prédéfinies

La version 6 de 4e Dimension a introduit les **constantes prédéfinies**. Ces constantes sont listées dans la fenêtre de l'Explorateur :



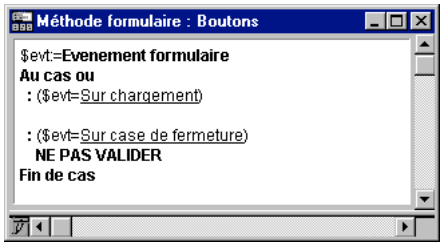
Les constantes prédéfinies sont listées par thèmes. Pour utiliser une constante prédéfinie dans la fenêtre de l'éditeur de méthodes, vous pouvez :
• soit glisser-déposer la constante depuis la fenêtre de l'Explorateur vers la fenêtre de l'éditeur de méthodes,
• soit saisir directement son nom dans la fenêtre de l'éditeur de méthodes.

Les noms des constantes prédéfinies peuvent contenir jusqu'à 31 caractères.

Astuce : Si vous saisissez directement le nom d'une constante prédéfinie, vous pouvez utiliser le symbole joker (@), ce qui vous évite d'avoir à taper le nom en entier. Par exemple, si vous saisissez “Données abs@”, 4D inscrira en entier la constante “Données absentes presse papiers” lorsque vous appuierez sur la touche Retour chariot ou Entrée pour valider la ligne de code.

Note : Les constantes prédéfinies (environ 500) sont listées par thèmes dans ce manuel. Reportez-vous à la section A propos de ce manuel pour plus d'informations. Lorsque c'est utile, les constantes prédéfinies sont également listées dans les descriptions des commandes.

Les constantes prédéfinies sont automatiquement soulignées dans la fenêtre de l'éditeur de méthodes et dans la fenêtre du débogueur :



Dans la fenêtre ci-dessus, Sur chargement est par exemple une constante prédéfinie.

Constantes littérales

4e Dimension accepte quatre types de données pour les constantes littérales :

- Chaîne,
- Numérique,
- Date,
- Heure.

Constantes chaînes

Une constante de type chaîne est incluse entre des guillemets droits ("..."). Voici quelques exemples de constantes chaîne :

"Ajouter Enregistrements"
"Aucun enregistrement trouvé."
"Facture"

Une chaîne vide est spécifiée par la succession de deux guillemets ("").

Constantes numériques

Une constante numérique s'écrit comme un nombre réel. Voici quelques exemple de constantes numériques :

27
123,76
0,0076

Les nombres négatifs s'écrivent précédés du signe moins (-). Par exemple:

–27
–123,76
–0,0076

Constantes dates

Une constante de type date est incluse entre deux points d'exclamation (!...!). Dans un environnement français, une date est structurée sous la forme jour/mois/année, un tiret (-) séparant les valeurs. Voici quelques exemples de constantes dates :

!1-1-76!
!4-4-04!
!25-12-92!

Une date nulle s'écrit !00-00-00!

Astuce : L'éditeur de méthodes dispose d'un raccourci pour entrer une date nulle. Pour cela, tapez un point d'exclamation (!) et appuyez sur la touche Entrée.

Note : Lorsqu'une année est saisie sur deux chiffres, 4e Dimension considère qu'elle appartient au 20e siècle, sauf si ce fonctionnement par défaut a été modifié à l'aide de la commande SIECLE PAR DEFAUT.

Constantes heures

Une constante heure est incluse entre deux points d'interrogation (?...?).

Note : Cette syntaxe peut être utilisée sous Windows et MacOS. Sous MacOS, vous pouvez également utiliser le symbole † (Option+t sur un clavier français).

Avec une version française de 4D, une heure est structurée sous la forme heure:minute:seconde, deux points (:) séparant les valeurs. Les heures sont stockées dans un format de 24 heures.

Voici quelques exemples de constantes heures :

?00:00:00? ` minuit
?09:30:00? ` 9:30 du matin
?13:01:59? ` 13 heures, 1 minute et 59 secondes

Une heure nulle s’écrit ?00:00:00?.

Astuce : L'éditeur de méthodes dispose d'un raccourci pour saisir une heure nulle. Pour cela, tapez un point d'interrogation (?) et appuyez sur la touche Entrée.

Référence

Conditions et boucles, Méthodes, Nommer les objets du langage 4D, Opérateurs, Pointeurs, Types de données, Variables.

Fondamentalement, dans 4e Dimension, les données peuvent être stockées de deux manières. Les champs stockent les données sur disque, de manière permanente ; les variables stockent les données en mémoire, de manière temporaire.

Lorsque vous définissez votre base, vous indiquez à 4e Dimension les noms et les types de champs que vous voulez utiliser. C'est pratiquement la même chose pour les variables — vous leur donnez un nom et un type.

Les types de variables suivants correspondent à chacun des types de données :
• Chaîne : chaîne alphanumérique fixe pouvant comprendre jusqu'à 255 caractères
• Texte : chaîne alphanumérique pouvant comprendre jusqu'à 32 000 caractères
• Entier : nombre entier compris entre -32768 et 32767
• Entier long : nombre entier compris entre -2^31 et (2^31) -1
• Réel (ou Numérique) : nombre réel compris entre ±1.7e±308 (15 chiffres)
• Date : date comprise entre 1/1/100 et 31/12/32767
• Heure : heure comprise entre 00:00:00 et 596000:00:00 (secondes depuis minuit)
• Booléen : Vrai ou Faux
• Image : toute image Windows ou MacOS
• BLOB (Binary Large OBject) : suite d'octets pouvant aller jusqu'à 2 Go
• Pointeur : pointeur vers une table, un champ, une variable, un tableau ou un élément de tableau

Vous pouvez afficher des variables à l'écran (à l'exception des pointeurs et des BLOB), les utiliser pour saisir des données, et les imprimer dans des états. Dans ces cas, elles se comportent exactement comme des champs, et les mêmes contrôles intégrés sont disponibles lorsque vous les créez :

• Formats d'affichage,
• Contrôles de saisie tels que filtres de saisie et valeurs par défaut,
• Filtrages des caractères,
• Enumérations (listes hiérarchiques)
• Valeurs saisissables ou non-saisissables

Les variables peuvent également servir à :

• contrôler des boutons (boutons, cases à cocher, boutons radio, boutons 3D, etc.),
• contrôler des thermomètres, règles et cadrans,
• contrôler des zones de défilement, des pop up menus et des listes déroulantes,
• contrôler des listes hiérarchiques et des menus hiérarchiques,
• contrôler des grilles de boutons, onglets, boutons image, etc.
• afficher les résultats de calculs ne devant pas être sauvegardés.

Créer des variables

Vous créez des variables simplement en les utilisant ; il n'est pas nécessaire de les déclarer formellement comme vous le faites avec les champs. Par exemple, si vous voulez créer une variable qui contient la date du jour plus 30 jours, il vous suffit d'écrire dans 4e Dimension :

```
MaDate:=Date du jour+30
```

MaDate est alors créée et contient la valeur que vous voulez. Le programme interprète la ligne comme "MaDate prend la valeur de la date courante plus 30 jours". Vous pourrez utiliser MaDate à chaque fois que vous le souhaiterez dans votre base. Par exemple, vous pouvez la stocker dans un champ du même type :

```
[MaTable]MonChamp:=MaDate
```

Il est parfois nécessaire de définir explicitement le type d'une variable. Pour plus d'informations sur le typage des variables dans une base que vous avez l'intention de compiler, reportez-vous à la section Commandes du thème compilateur.

Assigner des valeurs aux variables

Vous pouvez donner des valeurs aux variables et/ou récupérer leur valeur. Donner une valeur à une variable s'appelle **assigner une valeur** (ou **affecter une valeur**) et s'effectue à l'aide de l'opérateur d'assignation (:=). L'opérateur d'assignation est également utilisé pour assigner des valeurs aux champs.

L'opérateur d'assignation est le premier moyen pour créer une variable et lui donner une valeur. Vous placez le nom de la variable que vous voulez créer à gauche de l'opérateur. Par exemple :

```
MonNombre:=3
```

crée la variable MonNombre et lui donne la valeur numérique 3. Si MonNombre existait déjà, elle prend simplement la valeur 3.

Bien entendu, les variables ne seraient pas très utiles si vous ne pouviez pas récupérer les valeurs qu'elles contiennent. De nouveau, vous utilisez l'opérateur d'assignation. Si vous devez placer la valeur de MonNombre dans un champ nommé [Produits]Taille, il vous suffit de placer MonNombre à droite de l'opérateur d'assignation :

```
[Produits]Taille:=MonNombre
```

Dans ce cas, [Produits]Taille vaudrait 3. Cet exemple est plutôt simple, mais il illustre le moyen élémentaire dont vous disposez pour transférer des données d'un objet vers un autre en utilisant le langage.

Important : Ne confondez pas l'opérateur d'assignation (:=) avec le signe égal (=) qui est un opérateur de comparaison. L'assignation et la comparaison sont deux opérations très différentes. Pour plus d'informations sur les opérateurs de comparaison, reportez-vous à la section Opérateurs.

Variables locales, process et interprocess

Vous pouvez créer trois types de variables : des variables **locales**, des variables **process** et des variables **interprocess**. La différence entre ces trois types de variables est leur portée, ou les objets pour lesquels elles sont disponibles.

Variables locales

Le premier type de variable est la variable locale. Une variable locale, comme son nom l'indique, est locale à une méthode — c'est-à-dire accessible uniquement à l'intérieur de la méthode dans laquelle elle a été créée et inaccessible à l'extérieur de cette méthode. Pour une variable, être locale à une méthode signifie avoir une portée locale.

Vous utilisez une variable locale lorsque vous souhaitez limiter son fonctionnement à la méthode, pour une des raisons suivantes :

- Eviter des conflits de noms avec les autres variables.
- Utiliser temporairement des valeurs,
- Réduire le nombre de variables process.

Le nom d'une variable locale commence toujours par le signe dollar ($) et peut contenir jusqu'à 31 autres caractères. Si vous saisissez un nom plus long, 4e Dimension le tronque pour le ramener à 31 caractères.

Lorsque vous développez une base comportant de nombreuses méthodes et variables, il arrive souvent que vous n'ayez besoin d'utiliser une variable que dans une méthode. Vous pouvez alors créer et utiliser une variable locale, sans devoir vous soucier de l'existence d'une autre variable du même nom ailleurs dans la base.

Fréquemment, dans une base de données, des informations ponctuelles sont demandées à l'utilisateur. La commande Demander peut être appelée pour obtenir ces informations. Elle affiche une boîte de dialogue comportant un message demandant à l'utilisateur de répondre et, lorsque la réponse est validée, la retourne. Généralement, il n'est pas nécessaire de conserver cette information très longtemps dans vos méthodes. C'est l'endroit parfait pour utiliser une variable locale. Voici un exemple :

```
$vsID:=Demander("Saisissez votre numéro d'identification :")
Si (OK=1)
   CHERCHER ([Personnes];[Personnes]ID =$vsID)
Fin de si
```

Cette méthode demande simplement à l'utilisateur de saisir un numéro d'identification. La réponse est placée dans une variable locale, $vsID, puis la méthode la recherche parmi les champs [Personnes]ID. Une fois la méthode terminée, la variable locale $vsID est effacée de la mémoire. Ce fonctionnement est bien adapté puisque la variable n'est utile qu'une seule fois et dans cette méthode uniquement.

Variables process

Le second type de variable est la variable process. Une variable process est "visible" uniquement dans le process où elle a été créée. Elle est utilisable par toutes les méthodes du process et toutes les méthodes appelées depuis le process.

Le nom d'une variable process ne comporte aucun préfixe. Une variable process peut comporter jusqu'à 31 caractères.

En mode interprété, les variables sont gérées dynamiquement : elles sont créées en mémoire et effacées "à la volée". En mode compilé, tous les process que vous créez (process utilisateurs) partagent la même définition des variable process, mais chaque process dispose de sa propre instance pour chaque variable. Par exemple, la variable maVar est une certaine variable dans le process P_1 et une autre variable dans le process P_2.

A compter de la version 6, un process peut lire et écrire des variables process dans un autre process à l'aide des commandes LIRE VARIABLE PROCESS et ECRIRE VARIABLE PROCESS. Nous vous recommandons de n'utiliser ces commandes que dans le cadre des besoins décrits ci-dessous (qui sont les raisons pour lesquelles ces commandes ont été créées dans 4D) :
• Communication interprocess à des endroits particuliers de votre code
• Gestion du glisser-déposer interprocess
• En client/serveur, communication entre les process sur les postes clients et les procédures stockées exécutées sur le serveur.

Pour plus d'informations, reportez-vous à la section Introduction aux process et à la description de ces commandes.

Variables interprocess

Le troisième type de variable est la variable interprocess. Les variables interprocess sont visibles dans toute la base et sont disponibles pour tous les process.

Le nom d'une variable interprocess débute toujours par le symbole (<>) — formé du symbole "inférieur à" suivi du symbole "supérieur à" — et peut comporter jusqu'à 31 caractères supplémentaires.

Note : Cette syntaxe peut être utilisée sur les plates-formes Windows et Macintosh. En outre, sous MacOS uniquement, vous pouvez utiliser le symbole "diamant" (Option+v sur un clavier français).

Les variable interprocess sont principalement utilisées pour le partage d'informations entre les process.

En mode client/serveur, chaque poste (client et serveur) partage la même définition des variables interprocess, mais chacun utilise une instance différente d'une variable.

Variables objets dans les formulaires

Dans l'éditeur de formulaires, le fait de donner un nom à un objet actif — bouton, bouton radio, case à cocher, zone de défilement, thermomètre, etc. — crée automatiquement une variable de ce nom. Par exemple, si vous créez un bouton appelé MonBouton, une variable MonBouton est également créée. Notez bien que ce nom de variable n'est pas le libellé du bouton, mais son nom.

Ces variables vous permettent de contrôler et de gérer vos objets. Par exemple, lorsque l'utilisateur clique sur un bouton, la variable du bouton prend la valeur 1 ; sinon, le reste du temps, elle est à 0. La variable associée à un thermomètre ou un cadran vous permet de lire et de modifier les valeurs représentées. Par exemple, si l'utilisateur clique dans un thermomètre pour fixer une nouvelle valeur, la valeur de la variable est modifiée en conséquence. De même, si une méthode change la valeur de la variable, le thermomètre est redessiné pour afficher cette nouvelle valeur.

Pour plus d'informations sur les variables et les formulaires, reportez-vous au manuel *Mode Structure* de 4e Dimension ainsi qu'à la description de la fonction Evenement formulaire.

Variables système

4e Dimension exploite un certain nombre de variables particulières appelées **variables système**. Ces variables vous permettent de contrôler de nombreuses opérations. Toutes les variables système sont des variables process, disponibles à l'intérieur d'un seul process.

La plus importante de toutes est la variable système **OK**. Comme son nom le laisse supposer, elle vous indique si tout est OK dans un process particulier. Est-ce que l'enregistrement a bien été sauvegardé ? Est-ce que l'import d'enregistrements s'est bien déroulé ? Est-ce que l'utilisateur a cliqué sur le bouton OK ? La variable système OK prend la valeur 1 lorsqu'une opération s'est correctement déroulée, et 0 dans le cas contraire.

Pour plus d'informations sur les variables système, reportez-vous à la section traitant des Variables système.

Référence

Conditions et boucles, Constantes, Méthodes, Nommer les objets du langage 4D, Opérateurs, Pointeurs, Présentation des tableaux, Types de données.

4e Dimension gère un certain nombre de variables appelées **variables système**. Ces variables vous permettent de contrôler le déroulement de diverses opérations. Les variables systèmes sont toutes des variables process, accessibles uniquement à l'intérieur d'un process. Cette section décrit les variables système de 4e Dimension.

OK

La variable système OK est la plus couramment utilisée. En général, elle prend la valeur 1 lorsqu'une opération s'est correctement déroulée, et 0 lorsque l'opération a échoué. Les commandes suivantes modifient la valeur de la variable système OK :

Ajouter a document
AJOUTER ENREGISTREMENT
AJOUTER SOUS ENREGISTREMENT
BLOB VERS DOCUMENT
BLOB vers liste
BLOB VERS VARIABLE
CHANGER UTILISATEUR
CHERCHER
CHERCHER PAR EXEMPLE
CHERCHER PAR FORMULE DANS SELECTION
COMPRESSER BLOB
COPIER BLOB
Creer document
Creer fichier ressources
Demander
DESINSCRIRE CLIENT
DOCUMENT VERS BLOB
ECRIRE IMAGE DANS PRESSE PAPIERS
ECRIRE PROPRIETES RESSOURCE
ECRIRE RESSOURCE CHAINE
ECRIRE RESSOURCE TEXTE
ECRIRE VARIABLES
ECRITURE DIF
ENVOYER ENREGISTREMENT
ENVOYER PAQUET
ETAT
EXPORTER DONNEES
FIXER TAILLE BLOB
IMAGE VERS BLOB
IMPORTER DONNEES
IMPRIMER SELECTION
INSERER DANS BLOB
JOUER SON
LECTURE ASCII
AJOUTER A PRESSE PAPIERS
AJOUTER SEGMENT DE DONNEES
APPLIQUER A SELECTION
BLOB VERS IMAGE
BLOB vers texte
CHANGER PRIVILEGES
CHARGER ENSEMBLE
CHERCHER DANS SELECTION
CHERCHER PAR FORMULE
CHERCHER SUR CLE
CONFIRMER
COPIER DOCUMENT
CREER DOSSIER
DECOMPRESSER BLOB
DEPLACER DOCUMENT
DIALOGUE
ECRIRE FICHIER IMAGE
ECRIRE NOM RESSOURCE
ECRIRE RESSOURCE
ECRIRE RESSOURCE IMAGE
ECRIRE TEXTE DANS PRESSE PAPIERS
ECRITURE ASCII
ECRITURE SYLK
ENVOYER FICHIER HTML
ENVOYER VARIABLE
EXECUTER SUR CLIENT
FIXER HISTORIQUE
FIXER TIMEOUT
IMAGE VERS GIF
IMPRIMER ETIQUETTES
INSCRIRE CLIENT
JOINTURE
LANCER SERVEUR WEB
LECTURE DIF

LECTURE SYLK
LIRE CLIENTS INSCRITS
LIRE IMAGE DANS BIBLIOTHEQUE
LIRE PRESSE PAPIERS
Lire ressource chaine
LIRE RESSOURCE IMAGE
Lire texte dans presse papiers
LISTE DE CHAINES VERS TABLEAU
LISTE DES DOSSIERS
MODIFIER ENREGISTREMENT
NE PAS VALIDER
Ouvrir document
PARAMETRES IMPRESSION
RECEVOIR PAQUET
REGLER SERIE
Selectionner dossier
SUPPRIMER DANS BLOB
SUPPRIMER RESSOURCE
TABLEAU VERS LISTE DE CHAINES
TRIER
UTILISER FILTRE
VALIDER
VARIABLE VERS BLOB
Lire chaine dans liste
LIRE FICHIER IMAGE
LIRE IMAGE DANS PRESSE PAPIERS
LIRE RESSOURCE
LIRE RESSOURCE ICONE
Lire ressource texte
LIRE VARIABLES
LISTE DES DOCUMENTS
LISTE VERS BLOB
MODIFIER SOUS ENREGISTREMENT
Nom commande
Ouvrir fichier ressources
RECEVOIR ENREGISTREMENT
RECEVOIR VARIABLE
SELECTION RETOUR
STOCKER ENSEMBLE
SUPPRIMER DOCUMENT
TABLEAU VERS ENUMERATION
Taille document
TRIER PAR FORMULE
VALEURS DISTINCTES
VALIDER TRANSACTION

Document

La variable système Document contient soit le nom, soit le chemin d'accès et le nom (suivant la valeur passée en paramètre) du dernier fichier disque ayant été ouvert ou créé à l'aide d'une des commandes suivantes :

Ajouter a document
Creer document
ECRIRE FICHIER IMAGE
ECRITURE ASCII
ECRITURE SYLK
EXPORTER DONNEES
IMPORTER DONNEES
LECTURE ASCII
LECTURE SYLK
LIRE VARIABLES
Ouvrir fichier ressources
REGLER SERIE
CHARGER ENSEMBLE
Creer fichier ressources
ECRIRE VARIABLES
ECRITURE DIF
ETAT
FIXER HISTORIQUE
IMPRIMER ETIQUETTES
LECTURE DIF
LIRE FICHIER IMAGE
Ouvrir document
STOCKER ENSEMBLE
UTILISER FILTRE

FldDelimit

La variable système FldDelimit contient le code ASCII du caractère à utiliser comme délimiteur de champs lorsque vous importez ou exportez du texte. Par défaut, cette valeur est 9, c'est-à-dire le code ASCII du caractère Tabulation. Modifiez cette valeur pour changer de délimiteur de champs.

RecDelimit
La variable système RecDelimit contient le code ASCII du caractère à utiliser comme délimiteur d'enregistrements lorsque vous importez ou exportez du texte. Par défaut, cette valeur est 13, c'est-à-dire le code ASCII du caractère Retour chariot. Modifiez cette valeur pour changer de délimiteur d'enregistrements.

Error
La variable système Error n'est utilisable que dans une méthode installée par la commande APPELER SUR ERREUR. Cette variable contient le code de l'erreur. Les codes des erreurs de 4e Dimension et des erreurs Système sont listés dans les sections du thème “Codes d'erreurs”.

MouseDown, MouseX, MouseY, KeyCode, Modifiers et MouseProc
Ces variables système ne sont utilisables que dans une méthode installée par APPELER SUR EVENEMENT.

• La variable système MouseDown prend la valeur 1 si le bouton de la souris a été enfoncé. Sinon, elle prend la valeur 0.
• Si l'événement est un MouseDown (MouseDown=1), les variables système MouseX et MouseY contiennent les coordonnées verticale et horizontale de l'endroit où le clic a eu lieu. Les deux valeurs sont exprimées en pixels et avec le système de coordonnées locales de la fenêtre.
• La variable système KeyCode contient le ASCII de la touche ayant été enfoncée. Si la touche enfoncée était une touche de fonction, KeyCode contient un code spécial. Les codes ASCII et les codes des touches de fonction sont listés dans les sections Codes ASCII et Codes des touches de fonction.
• La variable système Modifiers contient les codes des *modifiers* du clavier (**Ctrl/Commande**, **Alt/Option**, **Maj**, **Verr. Maj**). Cette variable n'est significative que dans une méthode d'interruption sur événement installée par la commande APPELER SUR EVENEMENT.
• La variable système MouseProc contient le numéro du process dans lequel le dernier événement a eu lieu.

Référence
Présentation des ensembles, Variables.

Les pointeurs sont des outils de programmation avancée.

Lorsque vous utilisez le langage de 4e Dimension, vous vous référez aux différents objets par l'intermédiaire de leur nom — en particulier les tables, champs, variables et tableaux. Pour appeler l'un d'entre eux, vous écrivez simplement son nom. Cependant, il est parfois utile de pouvoir appeler ou référencer ces éléments sans nécessairement connaître leur nom. C'est ce que permettent les pointeurs.

Le concept de pointeur n'est pas tellement éloigné de la vie courante. Vous vous référez souvent à des choses sans connaître leur identité exacte. Par exemple, vous dites à un ami "Allons-y avec ta voiture" au lieu de "Allons-y avec la voiture immatriculée 123 ABD 99". Dans ce cas, vous faites référence à la voiture immatriculée 123 ABD 99 en utilisant l'expression "ta voiture". Par analogie, l'expression "la voiture immatriculée 123 ABD 99" est le nom d'un objet, et "ta voiture" est un pointeur référençant (ou pointant vers) l'objet.

La capacité de se référer à quelque chose sans connaître son identité exacte est très utile. Si votre ami s'achetait une nouvelle voiture, l'expression "ta voiture" serait toujours exacte — ce serait toujours une voiture et vous pourriez toujours aller quelque part avec. Les pointeurs fonctionnent de la même manière. Par exemple, un pointeur peut pointer à un moment donné vers un champ numérique appelé Age, et plus tard vers une variable numérique appelée Ancien âge. Dans les deux cas, le pointeur référence des données numériques pouvant être utilisée dans des calculs. Vous pouvez utiliser des pointeurs pour référencer des tables, des champs, des variables, des tableaux et des éléments de tableaux.

Le tableau suivant vous fournit un exemple de chaque type :

Objet	Référencement	Utilisation	Affectation
Table	vpTble:=->[Table]	TABLE DEFAUT(vpTble->)	n/a
Champ	vpChp:=->[Table]Chp	ALERTE(vpChp->)	vpChp->:="Jean"
Variable	vpVar:=->Variable	ALERTE(vpVar->)	vpVar->:="Jean"
Tableau	vpT:=->Tableau	TRIER TABLEAU(vpT->;>)	COPIER TABLEAU(Tab; vpT->)
Elém. tabl.	vpElem:=->Tableau{1}	ALERTE(vpElem->)	vpElem->:="Jean"

Utiliser des pointeurs : un exemple

Il est plus facile d'expliquer l'utilisation des pointeurs au travers d'un exemple. Cet exemple vous montre comment accéder à une variable par l'intermédiaire d'un pointeur. Nous commençons par créer la variable :

```
MaVar:="Bonjour"
```

MaVar est désormais une variable contenant la chaîne "Bonjour". Nous pouvons alors créer un pointeur vers MaVar :

```
MonPointeur:=->MaVar
```

Le symbole -> signifie "pointer vers" (ce symbole formé du caractère "tiret" (-) suivi du caractère "supérieur à"). Dans ce cas, il crée un pointeur qui référence ou "pointe vers" MaVar. Ce pointeur est assigné à MonPointeur via l'opérateur d'assignation.

MonPointeur est désormais une variable qui contient un pointeur vers MaVar. MonPointeur ne contient pas "Bonjour", la valeur de MaVar, mais vous pouvez utiliser MonPointeur pour obtenir la valeur contenue dans MaVar. L'expression suivante retourne la valeur de MaVar :

```
MonPointeur->
```

Dans ce cas, la chaîne "Bonjour" est retournée. Lorsque le symbole -> est placé derrière un pointeur, la valeur de l'objet vers lequel pointe le pointeur est récupérée. On dit alors qu'on **dépointe** le pointeur.

Il est important de comprendre que vous pouvez utiliser un pointeur suivi du symbole -> partout où vous auriez pu utiliser l'objet pointé lui-même. Vous pouvez placer l'expression MonPointeur-> partout où vous pourriez utiliser la variable originale MaVar.

Par exemple, l'instruction suivante affiche une boîte de dialogue d'alerte comportant le mot Bonjour :

```
ALERTE(MonPointeur->)
```

Vous pouvez également utiliser MonPointeur pour modifier la valeur de MaVar. Par exemple, l'instruction suivante stocke la chaîne "Au revoir" dans la variable MaVar :

```
MonPointeur->:="Au revoir"
```

Si vous examinez les deux utilisations de l'expression MonPointeur-> ci-dessus, vous constatez que cette expression se comporte exactement comme si vous aviez utilisé MaVar à sa place. En résumé : les deux lignes suivantes effectuent la même opération — elles affichent une boîte de dialogue d'alerte contenant la valeur courante de la variable MaVar :

```
ALERTE(MonPointeur->)
ALERTE(MaVar)
```

Les deux lignes suivantes effectuent la même opération ; elles assignent la chaîne "Au revoir" à MaVar :

```
MonPointeur->:="Au revoir"
MaVar:="Au revoir"
```

Utiliser des pointeurs vers des boutons

Ce paragraphe décrit l'utilisation d'un pointeur pour référencer un bouton. Un bouton (du point de vue du langage) n'est rien d'autre qu'une variable. Bien que les exemples de ce paragraphe utilisent des pointeurs pour référencer des boutons, les concepts présentés s'appliquent à l'utilisation de tout type d'objet pouvant être référencé par un pointeur.

Imaginons que vous disposiez dans vos formulaires d'un certain nombre de boutons devant être actifs ou inactifs. Chaque bouton est associé à une condition qui peut être VRAI ou FAUX. La condition indique s'il faut désactiver ou activer le bouton. Vous pouvez utiliser un test comme celui-ci chaque fois que vous devez désactiver ou activer le bouton :

```
Si (Condition)   ` Si la condition est VRAIE...
   ACTIVER BOUTON (MonBouton)   ` activer le bouton
Sinon         ` Dans l'autre cas...
   INACTIVER BOUTON (MonBouton)   ` désactiver le bouton
Fin de si
```

Vous pouvez avoir besoin, pour chaque bouton que vous avez créé, d'utiliser un test similaire, dans lequel seul le nom du bouton change. Afin d'être plus efficace, vous pouvez créer un pointeur pour référencer chaque bouton puis utiliser une sous-routine pour le test lui-même.

Vous devez utiliser des pointeurs si vous appelez une sous-routine, car il n'est pas possible de faire référence autrement aux variables des boutons. Par exemple, voici une méthode projet nommée FIXER_BOUTON, qui référence un bouton avec un pointeur :

```
` Méthode projet FIXER_BOUTON
` FIXER_BOUTON ( Pointeur ; Booléen )
` FIXER_BOUTON ( -> Bouton ; Activé ou Désactivé )
`
` $1 – Pointeur vers un bouton
` $2 – Booléen. Si VRAI, active le bouton. Si FAUX, désactive le bouton

Si ($2) ` Si la condition est VRAIE...
   ACTIVER BOUTON($1->) ` activer le bouton
Sinon ` Si ce n'est pas le cas...
   INACTIVER BOUTON ($1->) ` désactiver le bouton
Fin de si
```

Vous pouvez appeler la méthode projet FIXER_BOUTON de la manière suivante :

```
   ` ...
FIXER_BOUTON (->bValider;Vrai)
   ` ...
FIXER_BOUTON (->bValider;Faux)
   ` ...
FIXER_BOUTON (->bValider;([Employés]Nom#"")
   ` ...
Boucle ($vlRadioBouton;1;20)
   $vpRadioBouton:=Pointeur vers("r"+Chaine($vlRadioBouton))
   FIXER_BOUTON ($vpRadioBouton;Faux)
Fin de boucle
```

Utiliser des pointeurs vers des tables

Partout où le langage requiert un nom de table, vous pouvez utiliser un pointeur dépointé vers une table.
Pour créer un pointeur vers une table, écrivez une instruction du type :

```
TablePtr:=->[touteTable]
```

Vous pouvez également récupérer un pointeur vers une table à l'aide de la fonction Table. Par exemple :

```
TablePtr:=Table(20)
```

Vous pouvez utiliser le pointeur dépointé dans vos commandes, comme ceci :

```
TABLE PAR DEFAUT(TablePtr->)
```

Utiliser des pointeurs vers des champs

Partout où le langage requiert un nom de champ, vous pouvez utiliser un pointeur dépointé vers un champ.
Pour créer un pointeur vers un champ, écrivez une ligne d'instruction du type :

```
ChampPtr:=->[uneTable]CeChamp
```

Vous pouvez également récupérer un pointeur vers un champ à l'aide de la fonction Champ. Par exemple :

```
ChampPtr:=Champ(1; 2)
```

Vous pouvez utiliser le pointeur dépointé avec les commandes, comme ceci :

```
CHANGER JEU DE CARACTERES(ChampPtr->; "Arial")
```

Utiliser des pointeurs vers des éléments de tableau

Vous pouvez créer un pointeur vers un élément de tableau. Par exemple, les lignes d'instruction suivantes créent un tableau et assignent à une variable appelée ElémPtr un pointeur vers le premier élément :

```
TABLEAU REEL(unTableau; 10) ` Créer un tableau
ElémPtr:=->unTableau{1} ` Créer un pointeur vers l'élément de tableau
```

Vous pouvez alors utiliser le pointeur dépointé pour assigner une valeur à l'élément, comme ceci :

```
ElémPtr->:=8
```

Utiliser des pointeurs vers des tableaux

Vous pouvez créer un pointeur vers un tableau. Par exemple, les lignes d'instruction suivantes créent un tableau et assignent à la variable nommée TabPtr un pointeur vers le tableau :

```
TABLEAU REEL(unTableau; 10) ` Créer un tableau
TabPtr := ->unTableau  ` Créer un pointeur vers le tableau
```

Il est important de comprendre que ce pointeur pointe vers le tableau, et non vers un élément du tableau. Par exemple, vous pourriez utiliser le pointeur dépointé de la manière suivante :

```
TRIER TABLEAU(TabPtr->; >) ` Tri du tableau
```

Si vous devez vous référer au quatrième élément du tableau à l'aide du pointeur, vous pouvez écrire :

```
TabPtr->{4} := 84
```

Utiliser un tableau de pointeurs

Il est souvent utilise de disposer d'un tableau de pointeurs référençant un groupe d'objets homogène.

Un exemple d'utilisation typique d'un tel groupe d'objets est une grille de variables dans un formulaire. Chaque variable dans la grille est numérotée de manière séquentielle, par exemple : Var1, Var2,..., Var10. Vous devez souvent vous référer à ces variables, de manière indirecte — par leur numéro. Si vous créez un tableau de pointeurs et faites pointer les pointeurs vers chaque variable, il est alors facile de référencer les variables. Par exemple, pour créer un tableau et initialiser chaque élément, vous pouvez écrire :

```
TABLEAU POINTEUR(tabPointeurs; 10)  ` Créer un tableau de 10 pointeurs
Boucle ($i; 1; 10)  ` Boucle une fois par variable
   tabPointeurs{$i}:=Pointeur vers("Var"+Chaine($i))  ` Initialiser chaque élément du tableau
Fin de boucle
```

La fonction Pointeur vers retourne un pointeur vers l'objet passé en paramètre.

Pour référencer une des variables, il suffit d'appeler les éléments du tableau. Par exemple, pour remplir les variables avec les dix prochains jours (en supposant que les variables soient de type Date), vous pourriez écrire :

```
Boucle ($i; 1; 10) ` Boucle une fois par variable
   tabPointeurs{$i}->:=Date du jour+$i ` Assigner les jours
Fin de boucle
```

Sélectionner un bouton avec un pointeur

Si vous disposez d'un groupe homogène de boutons radio dans un formulaire, vous devrez souvent pouvoir définir rapidement leur état. La solution consistant à référencer chacun d'entre eux par son nom n'est pas très pratique. Imaginons que vous disposiez d'un groupe de cinq boutons radio libellés Bouton1, Bouton2,..., Bouton5.

Dans un groupe de boutons radio, seul l'un d'entre eux est sélectionné. Le numéro du bouton sélectionné peut être stocké dans un champ numérique. Par exemple, si le champ [Préférences]EtatBouton contient 3, alors Bouton3 est sélectionné. Dans la méthode formulaire, vous pouvez utiliser le code suivant pour paramétrer les boutons :

```
Au cas ou
   :(Evenement formulaire=Sur chargement)
      ` ...
      Au cas ou
         : ([Préférences]EtatBouton = 1)
            Bouton1:=1
         : ([Préférences]EtatBouton = 2)
            Bouton2:=1
         : ([Préférences]EtatBouton = 3)
            Bouton3:=1
         : ([Préférences]EtatBouton = 4)
            Bouton4:=1
         : ([Préférences]EtatBouton = 5)
            Bouton5:=1
      Fin de cas
      ` ...
Fin de cas
```

Un cas particulier doit être testé pour chaque bouton radio. La méthode peut être très longue si le formulaire contient de nombreux boutons radio. Heureusement, vous pouvez utiliser des pointeurs pour résoudre ce problème.

La fonction Pointeur vers vous permet de retourner un pointeur vers un bouton radio. L'exemple suivant utilise un pointeur de ce type pour référencer le bouton radio devant être sélectionné. Voici la méthode optimisée :

```
Au cas ou
   :(Evenement formulaire=Sur chargement)
      ` ...
      $vpRadio:=Pointeur vers("Bouton"+Chaine([Préférences]EtatBouton))
      $vpRadio->:=1
      ` ...
Fin de cas
```

Le numéro du bouton radio traité doit être stocké dans le champ [Préférences]EtatBouton. Cette opération peut être effectuée dans la méthode formulaire, pour l'événement formulaire Sur clic souris :

```
[Préférences]EtatBouton:=Bouton1+(Bouton2*2)+(Bouton3*3)+(Bouton4*4)+(Bouton5*5)
```

Passer des pointeurs aux méthodes

Vous pouvez passer un pointeur en tant que paramètre d'une méthode. A l'intérieur de la méthode, vous pouvez modifier l'objet référencé par le pointeur. Par exemple, la méthode suivante, RECOIT DEUX, reçoit deux paramètres qui sont des pointeurs. Elle passe l'objet référencé par le premier paramètre en caractères majuscules, et l'objet référencé par le second paramètre en caractères minuscules :

```
      ` Méthode projet RECOIT DEUX
      ` $1 – Pointeur vers un champ ou une variable de type Chaîne. Passe la chaîne en majuscules.
      ` $2 – Pointeur vers un champ ou une variable de type Chaîne. Passe la chaîne en minuscules.
   $1->:=Majusc($1->)
   $2->:=Minusc($2->)
```

L'instruction suivante emploie la méthode RECOIT DEUX pour passer un champ en caractères majuscules et une variable en caractères minuscules :

```
   RECOIT DEUX (->[MaTable]MonChamp; ->MaVar)
```

Si le champ, [MaTable]MonChamp, contenait la chaîne "dupont", celle-ci deviendrait "DUPONT". Si la variable MaVar contenait la chaîne "BONJOUR", celle-ci deviendrait "bonjour".

Dans la méthode RECOIT DEUX (et, en fait, à chaque fois que vous utilisez des pointeurs), il est important que les types de données des objets référencés soient corrects. Dans l'exemple précédent, les pointeurs doivent pointer vers des objets contenant une chaîne ou un texte.

Pointeurs vers des pointeurs

Si vous aimez compliquer les choses à l'extrême (bien que cela ne soit pas nécessaire dans 4e Dimension), vous pouvez utiliser des pointeurs pour référencer d'autres pointeurs. Examinons l'exemple suivant :

```
MaVar := "Bonjour"
PointeurUn := ->MaVar
PointeurDeux := ->PointeurUn
(PointeurDeux->)-> := "Au revoir"
ALERTE((PointeurDeux->)->)
```

Cet exemple affiche une boîte de dialogue d'alerte contenant "Au revoir".

Voici la description de chaque ligne de l'exemple :

• MaVar := "Bonjour"

→ Cette ligne place simplement la chaîne "Bonjour" dans la variable MaVar.

• PointeurUn := ->MaVar

→ PointeurUn contient désormais un pointeur vers MaVar.

• PointeurDeux :=->PointeurUn

→ PointeurDeux (une nouvelle variable) contient un pointeur vers PointeurUn, qui, elle, pointe vers MaVar.

• (PointeurDeux->)-> := "Au revoir"

→ PointeurDeux-> référence le contenu de PointeurUn, qui elle-même référence MaVar. Par conséquent, (PointeurDeux->)-> référence le contenu de MaVar. Donc, dans ce cas, la valeur "Au revoir" est assignée à la MaVar.

• ALERTE ((PointeurDeux->)->)

→ C'est ici la même chose que précédemment : PointeurDeux-> référence le contenu de PointeurUn, qui elle-même référence MaVar. Par conséquent, (PointeurDeux->)-> référence le contenu de MaVar. Donc, dans ce cas, la boîte de dialogue d'alerte affiche le contenu de maVar.

La ligne suivante place la valeur "Bonjour" dans MaVar :

```
(PointeurDeux->)->:="Bonjour"
```

La ligne suivante récupère "Bonjour" à partir de MaVar et la place dans NouvelleVar :

```
NouvelleVar:=(PointeurDeux->)->
```

Important : Vous devez utiliser des parenthèses lors des déréférencements multiples.

Référence

Conditions et boucles, Constantes, Méthodes, Nommer les objets du langage 4D, Opérateurs, Présentation des tableaux, Tableaux et pointeurs, Types de données, Variables.

Cette section décrit les conventions d'écriture employées pour les nombreux objets du langage de 4e Dimension. Le nom de chaque objet doit respecter les règles suivantes :

• Un nom doit commencer par un caractère alphabétique (une lettre).
• Le nom peut ensuite contenir des caractères alphabétiques, des caractères numériques, des espaces et des tirets bas (_).
• Les virgules, barres de fraction et deux points (:) sont interdits.
• Les caractères réservés car utilisés comme opérateurs, comme l'astérisque (*) ou le +, sont interdits.
• 4e Dimension ignore les espaces superflus.

Tables

Vous indiquez qu'un objet est une table en plaçant son nom entre crochets : [...]. Un nom de table peut contenir jusqu'à 31 caractères.

Exemples

```
TABLE PAR DEFAUT ([Commandes])
FORMULAIRE ENTREE ([Clients]; "Entrée")
AJOUTER ENREGISTREMENT ([Lettres])
```

Champs

Vous indiquez qu'un objet est un champ en spécifiant d'abord la table à laquelle il appartient. Le nom du champ se place immédiatement derrière celui de la table. Un nom de champ peut contenir jusqu'à 31 caractères.

Ne faites pas commencer un nom de champ par le tiret bas (_). Ce caractère est réservé pour l'utilisation des plug-ins. Lorsque 4e Dimension rencontre dans l'éditeur de méthodes un nom de champ débutant par le tiret bas, le caractère est supprimé.

Exemples

```
[Commandes]Total:=Somme([Ligne]Montant)
CHERCHER([Clients];[Clients]Nom="Dupont")
[Lettres]Texte:=Capitaliser texte ([Lettres]Texte)
```

Il n'est pas obligatoire, dans les méthodes, de faire figurer systématiquement le nom de la table à laquelle le champ appartient avant le nom du champ. Nous vous conseillons cependant de prendre l'habitude de le faire car c'est une technique de programmation efficace.

Sous-tables

Vous indiquez qu'un objet est une sous-table en spécifiant d'abord le nom de la table parente à laquelle elle appartient. Le nom de la sous-table se place immédiatement derrière celui de la table. Un nom de sous-table peut contenir jusqu'à 31 caractères.

Exemples

```
TOUS LES SOUS ENREGISTREMENTS ([Personnes]Enfants)
AJOUTER SOUS ENREGISTREMENT ([Clients]Tél;"Ajout")
SOUS ENREGISTREMENT SUIVANT ([Letters]Keywords)
```

Une sous-table est considérée comme un type de champ ; par conséquent, les mêmes règles que pour les champs lui sont appliquées lorsqu'elle est placée dans un formulaire. Il n'est pas obligatoire, dans les méthodes, de faire figurer systématiquement le nom de la table parente devant celui de la sous-table. Nous vous conseillons cependant de prendre l'habitude de le faire car c'est une bonne technique de programmation.

Sous-champs

Vous indiquez qu'un objet est un sous-champ de la même manière que pour un champ. Vous spécifiez d'abord le nom de la sous-table à laquelle il appartient, puis placez le nom du sous-champ immédiatement derrière. Les deux noms sont séparés par une apostrophe ('). Un nom de sous-champ peut contenir jusqu'à 31 caractères.

Exemples

```
[Personnes]Enfants'Prénom:=Majusc([Personnes]Enfants'Prénom)
[Clients]Tél'Numéro:="408 555 1212"
[Lettres]Mots-clés'Mot:=Capitaliser texte ([Lettres]Mots-clés'Mot)
```

Il n'est pas obligatoire, dans les méthodes objet ou formulaire de la sous-table, de faire figurer systématiquement les noms de la table parente et de la sous-table devant celui du sous-champ. Nous vous conseillons cependant de prendre l'habitude de le faire, car c'est une bonne technique de programmation.

Variables interprocess

Vous indiquez qu'un objet est une variable interprocess en faisant précéder son nom des symboles (<>), formés des caractères "inférieur à" suivi de "supérieur à".

Note : Cette syntaxe peut être utilisée sous Windows et MacOS. De plus, sous MacOS, vous pouvez également utiliser le caractère "diamant" (Option + v sur un clavier français).

Le nom d'une variable interprocess peut contenir jusqu'à 31 caractères, symbole <> non compris.

Exemples

```
<>vlProcessID:=Numero du process courant
<>vsKey:=Caractere(KeyCode)
Si (<>vtNom#"")
```

Variables process

Vous indiquez qu'un objet est une variable process en écrivant simplement son nom (qui ne doit pas commencer par les symboles $ ou <>). Ce nom peut contenir jusqu'à 31 caractères.

Exemples

```
vrGrandTotal:=Somme([Comptes]Montant)
Si (bValider=1)
vsNomCourant:=""
```

Variables locales

Vous indiquez qu'un objet est une variable locale en faisant précéder son nom du symbole dollar ($). Le nom d'une variable locale peut contenir jusqu'à 31 caractères, signe dollar non compris.

Exemples

```
Boucle ($vlEnregistrement; 1; 100)
Si ($vsTempVar="No")
$vsMaChaîne:="Bonjour à tous"
```

Tableaux

Vous indiquez qu'un objet est un tableau en écrivant simplement son nom, qui est celui que vous passez à une commande de déclaration de tableau (par exemple TABLEAU ENTIER LONG) lorsque vous créez le tableau. La tableaux sont des variables, et comme pour les variables, il existe trois types de tableaux qui se différencient par leur portée :

- Tableaux interprocess,
- Tableaux process,
- Tableaux locaux.

Tableaux interprocess
Le nom d'un tableau interprocess est précédé du symbole (<>), formé des caractères "inférieur à" suivi de "supérieur à".

Note : Cette syntaxe peut être utilisée sous Windows et MacOS. De plus, sous MacOS, vous pouvez également utiliser le caractère "diamant" (Option + v sur un clavier français).

Le nom d'un tableau interprocess peut contenir jusqu'à 31 caractères, symbole <> non compris.

Exemples

```
TABLEAU TEXTE(<>atSujets;Enregistrements dans table([Topics]))
TRIER TABLEAU (<>asMotsClés; >)
TABLEAU ENTIER(<>aiGrosTableau;10000)
```

Tableaux process
Vous indiquez qu'un objet est un tableau process en écrivant simplement son nom (qui ne doit pas commencer par les symboles $ ou <>). Ce nom peut contenir jusqu'à 31 caractères.

Exemples

```
TABLEAU TEXTE(atSujets;Enregistrements dans table([Topics]))
TRIER TABLEAU (asMotsClés; >)
TABLEAU ENTIER(aiGrosTableau;10000)
```

Tableaux locaux
Un tableau est déclaré local lorsque son nom est précédé du signe dollar ($). Le nom d'un tableau local peut contenir jusqu'à 31 caractères, signe dollar non compris.

Exemples

```
TABLEAU TEXTE($atSujets;Enregistrements dans table([Topics]))
TRIER TABLEAU ($asMotsClés; >)
TABLEAU ENTIER($aiGrosTableau;10000)
```

Eléments de tableaux
Vous désignez un élément d'un tableau local, process ou interprocess à l'aide d'accolades ({...}). L'élément référencé (l'indice) est indiqué par une expression numérique.

Exemples

```
   ` Adresser un élément d'un tableau interprocess
Si (<>asMotsClés{1}="Stop")
<>atSujets{$vlElem}:=[Topics]Sujet
$viValeurSuivante:=<>aiGrosTableau{Taille tableau(<>aiGrosTableau)}

   ` Adresser un élément d'un tableau process
Si (asMotsClés{1}="Stop")
atSujets{$vlElem}:=[Topics]Sujet
$viValeurSuivante:=aiGrosTableau{Taille tableau(aiGrosTableau)}

   ` Adresser un élément d'un tableau local
Si ($asMotsClés{1}="Stop")
$atSujets{$vlElem}:=[Topics]Sujet
$viValeurSuivante:=$aiGrosTableau{Taille tableau($aiGrosTableau)}
```

Eléments de tableaux à deux dimensions
Vous désignez un élément d'un tableau à deux dimensions à l'aide d'une double paire d'accolades ({...}). L'élément référencé (l'indice) est indiqué par deux expressions numériques dans deux paires d'accolades.

Exemples

```
   ` Adresser un élément d'un tableau interprocess à deux dimensions
Si (<>asMotsClés{$vlLigneSuivante}{1}="Stop")
<>atSujets{10}{$vlElem}:=[Topics]Sujet
$viValeurSuivante:=<>aiGrosTableau{$vlSet}{Taille tableau(<>aiGrosTableau{$vlSet})}

   ` Adresser un élément d'un tableau process à deux dimensions
Si (asMotsClés{$vlLigneSuivante}{1}="Stop")
atSujets{10}{$vlElem}:=[Topics]Sujet
$viValeurSuivante:=aiGrosTableau{$vlSet}{Taille tableau(aiGrosTableau{$vlSet})}
```

```
` Adresser un élément d'un tableau interprocess à deux dimensions
Si ($asMotsClés{$vlLigneSuivante}{1}="Stop")
$atSujets{10}{$vlElem}:=[Topics]Sujet
$viValeurSuivante:=$aiGrosTableau{$vlSet}{Taille tableau($aiGrosTableau{$vlSet})}
```

Formulaires

Vous indiquez qu'un objet est un formulaire en utilisant une expression de type chaîne alphanumérique qui représente son nom. Le nom d'un formulaire peut contenir jusqu'à 31 caractères.

Exemples

```
FORMULAIRE ENTREE([Personnes];"Entrée")
FORMULAIRE SORTIE([Personnes]; "Sortie")
DIALOGUE([Stock];"Boîte de note"+Chaine($vlStage))
```

Méthodes

Vous indiquez qu'un objet est une méthode (sous-routine ou fonction utilisateur) en saisissant son nom. Ce nom peut contenir jusqu'à 31 caractères.

Note : Une méthode qui ne retourne pas de résultat est appelée une procédure. Une méthode qui retourne un résultat est appelée une fonction utilisateur.

Exemples

```
Si (Nouveau client)
EFFACER VALEURS DUPLIQUEES
APPLIQUER A SELECTION ([Employés];AUGMENTER SALARIES)
```

Conseil : Nous vous recommandons d'adopter, pour nommer vos méthodes, la même convention que celle utilisée dans le langage de 4D : écrivez les noms de vos procédures en caractères majuscules, et vos fonctions en minuscules avec la première lettre en majuscule. Ainsi, lorsque vous réouvrirez une base au bout de plusieurs mois, vous identifierez immédiatement si une méthode retourne ou non un résultat, en regardant son nom dans la fenêtre de l'Explorateur.

Note : Lorsque vous souhaitez appeler une méthode, vous saisissez simplement son nom. Toutefois, certaines commandes intégrées telles que APPELER SUR ERREUR, ainsi que les commandes des plug-ins, nécessitent que vous passiez le nom d'une méthode en tant que chaîne lorsqu'un paramètre de type méthode est requis (cf. exemples ci-dessous).

Exemples

```
    ` Cette commande attend une méthode (fonction) ou une formule
CHERCHER PAR FORMULE ([aTable];Recherche Spéciale)
    ` Cette commande attend une méthode (procédure) ou une formule
APPLIQUER A SELECTION ([Employés];AUGMENTER SALARIES)
    ` Mais cette commande attend un nom de méthode
APPELER SUR EVENEMENT ("GERER EVENEMENTS")
    ` Et cette commnde de plug-in attend un nom de méthode
wr_Erreur ("WR GERER ERREURS")
```

Les méthodes peuvent accepter des paramètres (ou arguments). Le paramètres sont passés à la méthode entre parenthèses, à la suite du nom de la méthode. Les paramètres sont séparés par des points virgule (;). Les paramètres sont passés à la méthode appelée en tant que variables locales numérotées séquentiellement : $1, $2,..., $n. De plus, plusieurs paramètres consécutifs (s'ils sont les derniers) peuvent être adressés à l'aide de la syntaxe ${n} où n, expression numérique, est le numéro du paramètre.
A l'intérieur d'une fonction, la variable locale $0 contient la valeur à retourner.

Exemples

```
    ` Dans ELIMINER ESPACES, $1 est pointeur sur le champ [Personnes]Nom
ELIMINER ESPACES (->[Personnes]Nom)

    ` Dans Créateur tableau :
    ` - $1 est un numérique qui vaut 1
    ` - $2 est un numérique qui vaut 5
    ` - $3 est un texte ou un alpha qui vaut "Super"
    ` - La valeur résultante est assignée à $0
$vsRésultat:=Créateur tableau (1; 5; "Super")

    ` Dans Poubelle :
    ` - Les trois paramètres sont de type Texte ou Alpha
    ` - Vous pouvez y accéder par $1, $2 ou $3
    ` - Vous pouvez y accéder en écrivant, par exemple, ${$vlParam}
    ` où $vlParam vaut 1, 2 ou 3
    ` - La valeur résultante est assignée à $0
vtClone:=Poubelle ("est"; "le"; "il")
```

Commandes de plug-ins (procédures, fonctions et zones externes)

Vous indiquez qu'un objet est une commande de plug-in en écrivant son nom tel qu'il est défini dans le plug-in. Le nom d'une commande de plug-in peut contenir jusqu'à 31 caractères.

Exemples

WR SUPPRIMER SELECTION (wrZone)
$spNouvelleZone:=***sp_Hors ecran***

Ensembles

Dans 4e Dimension, il existe deux types d'ensembles qui se distinguent par leur portée :

• Ensembles interprocess,
• Ensembles process.

On peut également distinguer un troisième type d'ensemble, spécifique à 4D Server:
• Ensembles clients.

Ensembles interprocess
Un ensemble est déclaré interprocess lorsque son nom, qui est une expression de type chaîne alphanumérique, est précédé du symbole (<>), formé des caractères "inférieur à" suivi de "supérieur à".

Note : Cette syntaxe peut être utilisée sous Windows et MacOS. De plus, sous MacOS, vous pouvez également utiliser le caractère "diamant" (Option + v sur un clavier français).

Le nom d'un ensemble interprocess peut comporter jusqu'à 80 caractères, symbole <> non compris.

Ensembles process
Vous déclarez un ensemble process en passant simplement une expression de type chaîne qui représente son nom (et qui ne doit pas débuter par les symboles <> ou $). Le nom d'un ensemble process peut comporter jusqu'à 80 caractères.

Ensembles client
Le nom d'un ensemble client doit être précédé du symbole dollar ($). Ce nom peut comporter jusqu'à 80 caractères, symbole dollar non compris.

Note : Dans les versions précédentes de 4D Server, les ensembles étaient gérés par les postes clients sur lesquels ils avaient été créés. A partir de la version 6, les ensembles sont gérés par le serveur. Dans certains cas, pour des raisons particulières ou d'optimisation, vous pourrez avoir besoin d'utiliser des ensembles localement, sur les postes clients. Pour cela, il vous suffit de créer des ensembles client.

Exemples

```
   ` Ensembles interprocess
UTILISER ENSEMBLE("<>Enregistrements supprimés")
NOMMER ENSEMBLE([Clients];"<>Commandes clients")
Si (Enregistrements dans ensemble("<>Sélection"+Chaine($i))>0)
   ` Ensembles process
UTILISER ENSEMBLE("Enregistrements supprimés")
NOMMER ENSEMBLE([Clients];"Commandes clients")
Si (Enregistrements dans ensemble("Sélection"+Chaine($i))>0)
   ` Ensembles client
UTILISER ENSEMBLE("$Enregistrements supprimés")
NOMMER ENSEMBLE([Clients];"$Commandes clients")
Si (Enregistrements dans ensemble("$Sélection"+Chaine($i))>0)
```

Sélections temporaires

Dans 4e Dimension, il existe deux types de sélections temporaires, qui se distinguent par leur portée :

- Sélections temporaires interprocess,
- Sélections temporaires process.

Sélections temporaires interprocess
Vous désignez une sélection temporaire interprocess en utilisant une expression de type chaîne débutant par le symbole (<>), formé des caractères “inférieur à” suivi de “supérieur à”.

Note : Cette syntaxe peut être utilisée sous Windows et MacOS. De plus, sous MacOS, vous pouvez également utiliser le caractère “diamant” (Option + v sur un clavier français).

Le nom d'une sélection temporaire interprocess peut contenir jusqu’à 80 caractères, symbole <> non compris.

Sélections temporaires process
Vous déclarez une sélection temporaire process en passant simplement une expression de type chaîne qui représente son nom (et qui ne doit pas débuter par les symboles <> ou $). Le nom d’une sélection temporaire process peut comporter jusqu’à 80 caractères.

Exemples

```
   ` Sélection temporaire interprocess
UTILISER SELECTION([Clients];"<>ParCodePostal")
   ` Sélection temporaire process
UTILISER SELECTION([Clients];"ParCodePostal")
```

Process

En mode mono-utilisateur, ou sur le poste client en mode client/serveur, il existe deux types de process :

- Process globaux,
- Process locaux.

Process globaux
Vous déclarez un process global en passant une expression de type chaîne de caractères qui représente son nom (qui ne doit pas commencer par le symbole $). Le nom d'un process peut comporter jusqu'à 30 caractères.

Process locaux
Vous déclarez un process local lorsque son nom est précédé du symbole dollar ($). Le nom d'un process peut comporter jusqu'à 30 caractères, symbole dollar non compris.

Exemple

```
   ` Lancer le process global "Ajouter Clients"
$vlProcessID:=Nouveau process("P_AJOUT_CLIENTS";48*1024;"Ajouter Clients")
   ` Lancer le process local "$Suivre Souris"
$vlProcessID:=Nouveau process("P_SUIVRE_SOURIS";16*1024;"$Suivre Souris")
```

Résumé des conventions d'écriture dans 4D

Le tableau suivant résume les conventions utilisées par 4e Dimension pour nommer les objets dans les méthodes.

Objet	Long. max.	Exemple
Table	31	[Factures]
Champ	31	[Employés]Nom
Sous-table	31	[Amis]Enfants
Sous-champ	31	[Documents]Mots-clés'Mot-clé
Variable interprocess	<> + 31	<>vlProcessSuivantID
Variable process	31	vsNomCourant
Variable locale	$ + 31	$vlCompteurLocal
Formulaire	31	"Formulaire Web perso"
Tableau interprocess	<> + 31	<>apTableaux
Tableau process	31	asGenre
Tableau local	$ + 31	$atValeurs
Méthode	31	M_AJOUTER_CLIENTS
Commande de plug-in	31	wr_Inserer texte
Ensemble interprocess	<> + 80	"<>Enregistrements à archiver"

Ensemble process	80	"Enregistrements actuels"
Ensemble client	$ + 80	"$Sujets précédents"
Sélection temporaire	80	"Employés de A à Z"
Sélection temporaire interprocess	<> + 80	"<>Employés de Z à A"
Process local	$ + 30	"$SuivreEvénements"
Process global	30	"P_MODULE_FACTURES"

Résoudre les conflits de noms

Si un objet d'un certain type a le même nom qu'un autre objet d'un autre type (par exemple, si un champ est baptisé Personnes et qu'une variable est également nommée Personnes), 4e Dimension utilise un système de priorités pour identifier l'objet. Veillez à utiliser des noms uniques pour les différents éléments de votre base.

4e Dimension identifie les noms utilisés dans les méthodes en fonction de l'ordre de priorité suivant :

1. Champs
2. Commandes
3. Méthodes
4. Routines de plug-ins
5. Constantes prédéfinies
6. Variables

Par exemple, 4e Dimension dispose d'une fonction interne appelée Date. Si vous appelez Date une de vos méthodes, 4e Dimension considérera "Date" comme étant la fonction interne et non votre méthode. Vous ne pourrez pas appeler votre méthode. En revanche, si vous nommez un champ "Date", 4e Dimension considérera que vous souhaitez appeler votre champ et non la fonction intégrée.

Référence
Constantes, Méthodes, Opérateurs, Pointeurs, Présentation des tableaux, Types de données, Variables.

Quelle que soit la simplicité ou la complexité d'une méthode, vous utiliserez toujours un ou plusieurs types de structure de programmation. Les structures de programmation déterminent si et dans quel ordre les lignes d'instructions sont exécutées à l'intérieur d'une méthode. Il existe trois types de structures :

- séquentielle,
- conditionnelle,
- répétitive.

Le langage de 4e Dimension dispose d'instructions permettant de contrôler chacune de ces structures.

Structures séquentielles

Une structure séquentielle est une structure simple, linéaire. Une séquence est une série d'instructions que 4e Dimension exécute les unes après les autres, de la première à la dernière. Par exemple :

```
FORMULAIRE SORTIE ([Personnes]; "Listing")
TOUT SELECTIONNER ([Personnes])
VISUALISER SELECTION ([Personnes])
```

Une instruction d'une ligne, fréquemment utilisée pour les méthodes objet, est le cas le plus simple de structure séquentielle. Par exemple :

```
[Personnes]Nom:=Majusc([Personnes]Nom)
```

Structures conditionnelles

Une structure conditionnelle permet aux méthodes de tester une condition et d'exécuter des séquences d'instructions différentes en fonction du résultat. La condition est une expression booléenne, c'est-à-dire pouvant retourner VRAI ou FAUX. L'une des structures conditionnelles est la structure Si...Sinon...Fin de si, qui aiguille le déroulement du programme vers une séquence ou une autre. L'autre structure conditionnelle est la structure Au cas ou...Sinon...Fin de cas, qui aiguille le programme vers une séquence parmi une ou plusieurs alternatives.

Structures répétitives (ou "boucles")

Il est très courant, lorsque vous écrivez des méthodes, de rencontrer des cas où vous devez répéter une séquence d'instructions un certain nombre de fois. Pour traiter ces besoins, le langage vous propose trois structures répétitives : Boucle...Fin de boucle, Tant que...Fin tant que, et Repeter...Jusque. Les boucles sont contrôlées de deux manières : soit elles se répètent jusqu'à ce qu'une condition soit remplie, soit elles se répètent un nombre fixé de fois. Chaque structure répétitive peut être utilisée de l'une ou l'autre manière, mais les boucles Tant que et Repeter sont mieux adaptées à la répétition jusqu'à ce qu'une condition soit remplie, alors que les boucles Boucle sont mieux adaptées à la répétition un certain nombre de fois.

Référence

Méthodes, Opérateurs logiques.

La syntaxe de la structure conditionnelle Si...Sinon...Fin de si est la suivante :

```
Si (Expression_booléenne)
   instruction(s)
Sinon
   instruction(s)
Fin de si
```

A noter que l'élément Sinon est optionnel, vous pouvez écrire :

```
Si (Expression_booléenne)
   instruction(s)
Fin de si
```

La structure Si...Sinon...Fin de si permet à votre méthode de choisir entre deux alternatives, en fonction du résultat, VRAI ou FAUX, d'un test (une expression booléenne).

Si l'expression booléenne est VRAIE, les instructions qui suivent immédiatement le test sont exécutées. Si l'expression booléenne est FAUSSE, les instructions suivant la ligne Sinon sont exécutées. Le Sinon est optionnel ; lorsqu'il est omis, c'est la première ligne d'instructions suivant le Fin de si (s'il y en a une) qui est exécutée.

Exemple

```
   ` Demander à l'utilisateur de saisir un nom
$Rech:=Demander("Saisissez un nom :")
Si (OK=1)
   CHERCHER([Personnes]; [Personnes]Nom=$Rech)
Sinon
   ALERTE("Vous n'avez pas saisi de nom.")
Fin de si
```

Astuce : Il n'est pas obligatoire que des instructions soient exécutées dans chacune des deux alternatives. Lorsque vous développez un algorithme, ou lorsque vous poursuivez un but précis, rien ne vous empêche d'écrire :

```
Si (Expression_booléenne)
Sinon
   instruction(s)
Fin de si
```

ou :

```
Si (Expression_booléenne)
   instruction(s)
Sinon
Fin de si
```

Référence

Au cas ou...Sinon...Fin de cas, Boucle...Fin de boucle, Conditions et boucles, Repeter...Jusque, Tant que...Fin tant que.

La syntaxe de la structure conditionnelle Au cas ou...Sinon...Fin de cas est la suivante :

```
Au cas ou
   : (Expression_booléenne)
      instruction(s)
   : (Expression_booléenne)
      instruction(s)
   .
   .
   .

   : (Expression_booléenne)
      instruction(s)
Sinon
   instruction(s)
Fin de cas
```

A noter que l'élément Sinon est optionnel, vous pouvez donc écrire :

```
Au cas ou
   : (Expression_booléenne)
      instruction(s)
   : (Expression_booléenne)
      instruction(s)
   .
   .
   .

   : (Expression_booléenne)
      instruction(s)
Fin de cas
```

Tout comme la structure Si...Sinon...Fin de si, la structure Au cas ou...Sinon...Fin de cas permet également à votre méthode de choisir parmi plusieurs séquences d'instructions. A la différence de la structure précédente, le Au cas ou...Sinon...Fin de cas peut tester un nombre illimité d'expressions booléennes et exécuter la séquence d'instructions correspondant à la valeur VRAI.

Chaque expression booléenne débute par le caractère deux points (:). La combinaison de deux points et d'une expression booléenne est appelé un cas. Par exemple, la ligne suivante est un cas :

```
: (bValider=1)
```

Seules les instructions suivant le premier cas VRAI (et ce, jusqu'au cas suivant) seront exécutées. Si aucun des cas n'est VRAI, aucune instruction n'est exécutée (s'il n'y a pas d'élément Sinon).

Vous pouvez placer une instruction Sinon après le dernier cas. Si tous les cas sont FAUX, les instructions suivant le Sinon seront exécutées.

Exemple
Cet exemple teste une variable numérique et affiche une boîte de dialogue d'alerte comportant un simple mot :

```
Au cas ou
   : (vRésult = 1)   ` Test si le numéro est 1
      ALERTE("Un.")   ` Si c'est 1, afficher une alerte
   : (vRésult = 2)   ` Test si le numéro est 2
      ALERTE("Deux.")   ` Si c'est 2, afficher une alerte
   : (vRésult = 3) ` Test si le numéro est 3
      ALERTE("Trois.")   ` Si c'est 3, afficher une alerte
Sinon   ` Si ce n'est ni 1 ni 2 ni 3, afficher une alerte
   ALERTE("Ce n'est ni un, ni deux, ni trois.")
Fin de cas
```

A titre de comparaison, voici la version avec Si...Sinon...Fin de si de la même méthode :

```
Si (vRésult = 1)    ` Test si le numéro est 1
   ALERTE ("Un.")    ` Si c'est 1, afficher une alerte
Sinon
   Si (vRésult = 2)    ` Test si le numéro est 2
      ALERTE ("Deux.")    ` Si c'est 2, afficher une alerte
   Sinon
      Si (vRésult = 3)    ` Test si le numéro est 3
         ALERTE ("Trois.")    ` Si c'est 3, afficher une alerte
      Sinon    ` Si ce n'est ni 1, 2 ni 3, afficher l'alerte
         ALERTE ("Ce n'est ni un, ni deux, ni trois.")
      Fin de si
   Fin de si
Fin de si
```

Rappelez-vous qu'avec une structure de type Au cas ou...Sinon...Fin de cas, seul le premier cas VRAI rencontré est exécuté. Même si d'autres cas sont VRAIS, seules les instructions suivant le premier cas VRAI seront prises en compte.

Par conséquent, lorsque vous testez dans la même méthode des cas simples et des cas complexes, vous devez placer les cas complexes **avant** les cas simples, sinon ils ne seront jamais exécutés.
Par exemple, si vous souhaitez traiter le cas simple (vRésult=1) et le cas complexe (vRésult=1) & (vDemande#2) et que vous structurez la méthode de la manière suivante :

```
Au cas ou
   : (vRésult = 1)
      ...   `instruction(s)
   : ((vRésult = 1) & (vDemande#2))   `← Les instructions suivant ce cas ne seront
                                      ` JAMAIS exécutées
      ...   `instruction(s)
Fin de cas
```

... les instructions associées au cas complexe ne seront jamais exécutées. En effet, pour que ce cas soit VRAI, ses deux conditions booléennes doivent l'être. Or, la première condition est celle du cas simple situé précédemment. Lorsqu'elle est VRAIE, le cas simple est exécuté et 4D sort de la structure conditionnelle, sans évaluer le cas complexe. Pour que ce type de méthode fonctionne, vous devez écrire :

```
Au cas ou
   : (vRésult = 1) & (vDemande#2)   `← Les cas complexes doivent toujours être placés
                                    ` en premier
      ...   `Instruction(s)
   : (vRésult = 1)
      ...   `Instruction(s)
Fin de cas
```

Astuce

Il n'est pas obligatoire que des instructions soient exécutées dans toutes les alternatives. Lorsque vous développez un algorithme, ou lorsque vous poursuivez un but particulier, rien ne vous empêche d'écrire :

```
Au cas ou
   : (Expression_booléenne)
   : (Expression_booléenne)

   .
   .
   .

   : (Expression_booléenne)
      instruction(s)
Sinon
   instruction(s)
Fin de cas
```

ou :

```
Au cas ou
   : (Expression_booléenne)
   : (Expression_booléenne)
      instruction(s)
   .
   .
   .

   : (Expression_booléenne)
      instruction(s)
Sinon
Fin de cas
```

ou :

```
Au cas ou
Sinon
   instruction(s)
Fin de cas
```

Référence

Boucle...Fin de boucle, Conditions et boucles, Repeter...Jusque, Si...Sinon...Fin de si, Tant que...Fin tant que.

La syntaxe de la structure répétitive (ou boucle) Tant que...Fin tant que est la suivante :

```
Tant que (Expression_booléenne)
   instruction(s)
Fin tant que
```

Une boucle Tant que exécute les instructions comprises entre Tant que et Fin tant que aussi longtemps que l'expression booléenne est VRAIE. Elle teste l'expression booléenne initiale et n'entre pas dans la boucle (et donc n'exécute aucune instruction) si l'expression est à FAUX.

Il est utile d'initialiser la valeur testée dans l'expression booléenne juste avant d'entrer dans la boucle Tant que. Initialiser la valeur signifie lui affecter un contenu approprié, généralement pour que l'expression booléenne soit VRAIE et que le programme entre dans la boucle.

La valeur de l'expression booléenne doit pouvoir être modifiée par un élément situé à l'intérieur de la boucle, sinon elle s'exécutera indéfiniment. La boucle suivante est sans fin car Infini est toujours VRAI :

```
Infini:=Vrai
Tant que (Infini)
Fin tant que
```

Si vous vous retrouvez dans une telle situation (où une méthode s'exécute de manière incontrôlée), vous pouvez utiliser les fonctions de débogage de 4D et remonter à la source du problème. Pour plus d'informations sur ce point, reportez-vous à la section Débogage.

Exemple

```
   ` Est-ce que l'utilisateur veut ajouter un enregistrement?
CONFIRMER ("Ajouter un enregistrement?")
Tant que (OK = 1)   ` Tant que l'utilisateur accepte
   AJOUTER ENREGISTREMENT([Table])   ` Ajouter un nouvel enregistrement
Fin tant que   ` Une boucle Tant que se termine toujours par Fin tant que
```

Dans cet exemple, la valeur de la variable système OK est définie par la commande CONFIRMER avant que le programme n'entre dans la boucle. Si l'utilisateur clique sur le bouton OK dans la boîte de dialogue de confirmation, la variable OK prend la valeur 1 et la boucle est exécutée. Dans le cas contraire, la variable OK prend la valeur 0 et la boucle est ignorée. Une fois que le programme entre dans la boucle, la commande AJOUTER ENREGISTREMENT permet de continuer à l'exécuter car elle met la variable système OK à 1 lorsque l'utilisateur sauvegarde l'enregistrement.
Lorsque l'utilisateur annule (ne valide pas) le dernier enregistrement, la variable système OK prend la valeur 0 et la boucle s'arrête.

Référence

Au cas ou...Sinon...Fin de cas, Boucle...Fin de boucle, Conditions et boucles, Repeter...Jusque, Si...Sinon...Fin de si.

La syntaxe de la structure répétitive (ou boucle) Repeter...Jusque est la suivante :

```
Repeter
   instruction(s)
Jusque (Expression_booléenne)
```

La boucle Repeter...Jusque est semblable à la boucle Tant que...Fin tant que, à la différence près qu'elle teste la valeur de l'expression booléenne après l'exécution de la boucle et non avant. Ainsi, la boucle est toujours exécutée au moins une fois, tandis que si l'expression booléenne est initialement à FAUX, la boucle Tant que...Fin tant que ne s'exécute pas du tout.

L'autre particularité de la boucle Repeter...Jusque est qu'elle se poursuit jusqu'à ce que l'expression booléenne soit à VRAI.

Exemple

Comparez l'exemple suivant avec celui de la boucle Tant que...Fin tant que : vous constatez qu'il n'est pas nécessaire d'initialiser l'expression booléenne — il n'y a pas de commande CONFIRMER pour initialiser la variable OK.

```
Repeter
   AJOUTER ENREGISTREMENT([aTable])
Jusque (OK=0)
```

Référence

Au cas ou...Sinon...Fin de cas, Boucle...Fin de boucle, Conditions et boucles, Si...Sinon...Fin de si, Tant que...Fin tant que.

La syntaxe de la structure répétitive Boucle...Fin de boucle est la suivante :

Boucle (Variable_Compteur; Expression_Début;Expression_Fin {; Expression_Incrément})
instructions(s)
Fin de boucle

La structure Boucle...Fin de boucle est une boucle contrôlée par un compteur :
• La variable compteur Variable_Compteur est une variable numérique (Réel, Entier ou Entier long) initialisée par Boucle...Fin de boucle à la valeur spécifiée par Expression_Début.
• La variable Variable_Compteur est incrémentée de la valeur spécifiée par le paramètre optionnel Expression_Incrément à chaque fois que la boucle est exécutée. Si vous ne passez pas de valeur dans Expression_Incrément, la variable compteur est incrémentée par défaut de un (1).
• Lorsque le compteur atteint la valeur définie par Expression_Fin, la boucle s'arrête.

Important : Les expressions numériques Expression_Début, Expression_Fin et Expression_Fin sont évaluées une seule fois, au début de la boucle. Si ces expressions sont des variables, leur modification depuis l'intérieur de la boucle **n'affectera pas l'exécution de la boucle**.

Astuce : En revanche, vous pouvez, si vous le souhaitez, modifier la valeur de la variable Variable_Compteur depuis l'intérieur de la boucle et cela **affectera l'exécution de la boucle**.

• Généralement, Expression_Début est inférieure à Expression_Fin.
• Si les deux expressions sont égales, la boucle ne sera exécutée qu'une fois.
• Si Expression_Début est supérieure à Expression_Fin, la boucle ne s'exécutera pas du tout, à moins que vous ne spécifiiez une Expression_Incrément négative. Reportez-vous ci-dessous au paragraphe décrivant ce point.

Exemples élémentaires

(1) La boucle suivante s'exécute 100 fois :

Boucle (vCompteur;1;100)
` Faire quelque chose
Fin de boucle

(2) L'exemple suivant permet de traiter tous les éléments du tableau unTableau :

```
Boucle ($vlElem;1;Taille tableau(unTableau))
      ` Faire quelque chose avec l'élément
   unTableau{$vlElem}:=...
Fin de boucle
```

(3) L'exemple suivant permet d'examiner chaque caractère du texte vtDuTexte :

```
Boucle ($vlCar;1;Longueur(vtDuTexte))
      ` Faire quelque chose avec le caractère si c'est une tabulation
   Si (Code ascii(vtDuTexte[[$vlCar]])=Caractere(Tab))
      ` ...
   Fin de si
Fin de boucle
```

(4) L'exemple suivant permet de traiter tous les enregistrements de la sélection de la table [uneTable]:

```
DEBUT SELECTION([uneTable])
Boucle ($vlEnrg;1;Enregistrements trouves([uneTable]))
      ` Faire quelque chose avec chaque enregistrement
   ENVOYER ENREGISTREMENT([uneTable])
      ` ...
      ` Passer à l'enregistrement suivant
   ENREGISTREMENT SUIVANT([uneTable])
Fin de boucle
```

La plupart des structures Boucle...Fin de boucle que vous écrirez dans vos bases ressembleront à celles présentées ci-dessus.

Décrémenter la variable Compteur

Dans certains cas, vous pouvez souhaiter disposer d'une boucle dont la valeur de la variable compteur décroît au lieu de croître. Pour cela, Expression_Début doit être supérieure à Expression_Fin et Expression_Increment doit être négative. Les exemples suivants effectuent les mêmes tâches que les précédents, mais en sens inverse :

(5) La boucle suivante s'exécute 100 fois :

```
Boucle (vCompteur;100;1;-1)
   ` Faire quelque chose
Fin de boucle
```

(6) L'exemple suivant permet de traiter tous les éléments du tableau unTableau :

```
Boucle ($vlElem;Taille tableau(unTableau);1;-1)
      ` Faire quelque chose avec l'élément
   unTableau{$vlElem}:=...
Fin de boucle
```

(7) L'exemple suivant permet d'examiner chaque caractère du texte vtDuTexte :

```
Boucle ($vlCar;Longueur(vtDuTexte);1;-1)
      ` Faire quelque chose avec le caractère si c'est une tabulation
   Si (Code ascii(vtDuTexte[[$vlCar]])=Caractere(Tab))
      ` ...
   Fin de si
Fin de boucle
```

(8) L'exemple suivant permet de traiter tous les enregistrements de la sélection de la table [uneTable]:

```
ALLER A DERNIER ENREGISTREMENT([uneTable])
Boucle ($vlEnrg;Enregistrements trouves([uneTable]);1;-1)
      ` Faire quelque chose avec chaque enregistrement
   ENVOYER ENREGISTREMENT([uneTable])
      ` ...
      ` Passer à l'enregistrement précédent
   ENREGISTREMENT PRECEDENT([uneTable])
Fin de boucle
```

Incrementer la variable compteur de plus de 1

Si vous le souhaitez, vous pouvez passer dans Expression_Incrément une valeur (positive ou négative) dont la valeur absolue est supérieure à un.

(9) La boucle suivante ne traite que les éléments pairs du tableau unTableau :

```
Boucle ($vlElem;2;((Taille tableau(unTableau)+1)\2)*2;2)
      ` Faire quelque chose avec l'élément 2,4...2n
   unTableau{$vlElem}:=...
Fin de boucle
```

Notez l'expression ((Taille tableau(unTableau)+1)\2)*2 qui permet de traiter indifféremment les tableaux de taille paire et impaire.

Sortir d'une boucle en modifiant la variable compteur

Dans certains cas, vous voudrez exécuter une boucle un certain nombre de fois, mais également pouvoir sortir si une autre condition devient Vraie.

Pour cela, il vous suffit de tester cette condition à l'intérieur de la boucle et, si elle devient Vraie, de "forcer" la valeur de la variable compteur, de manière à ce qu'elle soit supérieure à celle de Expression_Fin.

(10) Dans l'exemple suivant, vous effectuez une boucle parmi les enregistrements d'une sélection jusqu'à ce que la fin de la sélection soit atteinte, ou bien jusqu'à ce que la variable interprocess <>vbStop, initialement fixée à Faux, prenne la valeur Vrai. Cette variable est gérée par une méthode projet APPELER SUR EVENEMENT utilisée pour interrompre l'opération :

```
<>vbStop:=Faux
APPELER SUR EVENEMENT ("GESTION STOP")
   ` GESTION STOP définit <>vbStop à Vrai si les touches Ctrl+point (Windows) ou
   ` Commande+point (MacOS) sont enfoncées
$vlNbEnrgs:=Enregistrements trouves([aTable])
DEBUT SELECTION([aTable])
Boucle ($vlEnrgs;1;$vlNbEnrgs)
      ` Faire quelque chose avec l'enregistrement
   ENVOYER ENREGISTREMENT([aTable])
      ` ...
      ` Aller à l'enregistrement suivant
   Si (<>vbStop)
      $vlEnrgs:=$vlNbEnrgs+1 ` Forcer la variable compteur à stopper la boucle
   Sinon
      ENREGISTREMENT SUIVANT([aTable])
   Fin de si
Fin de boucle
APPELER SUR EVENEMENT("")
Si (<>vbStop)
   ALERTE("L'opération a été interrompue.")
Sinon
   ALERTE("L'opération s'est terminée avec succès.")
Fin de si
```

Comparaison des structures répétitives

Reprenons le premier exemple employé pour la structure Boucle...Fin de boucle :

(1) La boucle suivante s'exécute 100 fois :

```
Boucle (vCompteur;1;100)
   ` Faire quelque chose
Fin de boucle
```

Il est intéressant d'examiner la manière dont les boucles Tant que...Fin tant que et Repeter...Jusque effectuent la même action :

Voici la boucle Tant que...Fin tant que équivalente :

```
$i := 1 ` Initialisation du compteur
Tant que ($i<=100) ` Boucle 100 fois
   ` Faire quelque chose
   $i := $i + 1 ` Il faut incrémenter le compteur
Fin tant que
```

Voici la boucle Repeter...Jusque équivalente :

```
$i := 1 ` Initialisation du compteur
Repeter
   ` Faire quelque chose
   $i := $i + 1 ` Il faut incrémenter le compteur
Jusque ($i=100) ` Boucle 100 fois
```

Astuce : La boucle Boucle...Fin de boucle est généralement plus rapide que les boucles Tant que...Fin tant que et Repeter...Jusque car 4e Dimension teste la condition en interne pour chaque cycle de la boucle et incrémente lui-même le compteur. Par conséquent, nous vous conseillons de préférer à chaque fois que c'est possible la structure Boucle...Fin de boucle.

Optimiser l'exécution de Boucle...Fin de boucle

Vous pouvez utiliser comme compteur une variable interprocess, process ou locale, et lui attribuer le type Réel, Entier ou Entier long. Pour des boucles longues, et particulièrement en mode compilé, nous vous conseillons d'employer des variables locales de type Entier long.

(11) Voici un exemple :

```
C_ENTIER LONG($vlCompteur)   ` Utilisons une variable locale de type Entier long
Boucle  ($vlCompteur;1;10000)
   ` Faire quelque chose
Fin de boucle
```

Structures Boucle...Fin de boucle **emboîtées**

Vous pouvez emboîter autant de structures répétitives que vous voulez (dans les limites du raisonnable). Cela s'applique aux structures de type Boucle...Fin de boucle. Il y a dans ce cas une erreur courante à éviter : assurez-vous d'utiliser une variable compteur différente par structure de boucle. Voici deux exemples :

(12) Cet exemple permet de traiter tous les éléments d'un tableau à deux dimensions :

```
Boucle  ($vlElem;1;Taille tableau(unTableau))
      ` ...
      ` Faire quelque chose avec la ligne
      ` ...
   Boucle  ($vlSousElem;1;Taille tableau(unTableau{$vlElem}))
      ` Faire quelque chose avec l'élément
      unTableau{$vlElem}{$vlSousElem}:=...
   Fin de boucle
Fin de boucle
```

(13) L'exemple suivant construit un tableau de pointeurs vers tous les champs de type Date présents dans la base :

```
TABLEAU POINTEUR($apChampsDate;0)
$vlElem:=0
Boucle  ($vlTable;1;Nombre de tables)
   Boucle($vlChamp;1;Nombre de champs($vlTable))
      $vpChamp:=Champ($vlTable;$vlChamp)
      Si  (Type($vpChamp>)=Est une date)
         $vlElem:=$vlElem+1
         INSERER LIGNES($apChampsDate;$vlElem)
         $apChampsDate{$vlElem}:=$vpChamp
      Fin de si
   Fin de boucle
Fin de boucle
```

Référence

Au cas ou...Sinon...Fin de cas, Conditions et boucles, Repeter...Jusque, Si...Sinon...Fin de si, Tant que...Fin tant que.

Pour faire fonctionner les commandes, les opérateurs, et les autres composants du langage, vous les placez dans des méthodes. Ce chapitre décrit les fonctionnalités communes à tous les types de méthodes. Il existe plusieurs types de méthodes : les méthodes objet, les méthodes formulaire, les méthodes table (ou triggers), les méthodes projet et les méthodes base.

Une méthode est composée de plusieurs **lignes d'instructions**. Une ligne d'instruction effectue une action. Cette ligne d'instruction peut être simple ou complexe. Cette ligne peut être aussi longue que vous voulez (elle peut comporter jusqu'à 32 000 caractères, ce qui est normalement suffisant pour la plupart des instructions).

Par exemple, la ligne suivante est une instruction qui ajoute un nouvel enregistrement à la table [Personnes] :

```
AJOUTER ENREGISTREMENT([Personnes])
```

Une méthode contient également des **tests** et des **boucles** qui structurent son exécution. Pour plus d'informations sur les structures de programmation, reportez-vous à la section Conditions et boucles.

Types de méthodes

Il existe cinq types de méthodes dans 4e Dimension :

• **Méthodes objet** : une méthode objet est une courte méthode associée à un objet actif dans un formulaire. En général, les méthodes objet "gèrent" l'objet au moment de l'affichage ou de l'impression du formulaire. Vous ne pouvez pas appeler une méthode objet, 4D l'exécute automatiquement lorsqu'un événement implique l'objet auquel la méthode est rattachée.

• **Méthodes formulaire** : Une méthode formulaire est associée à un formulaire. Vous pouvez utiliser une méthode formulaire pour gérer les données et les objets, mais il est généralement plus simple et plus efficace d'utiliser des méthodes objet dans ce cas. Vous ne pouvez pas appeler une méthode formulaire, 4D gère automatiquement son exécution lorsqu'un événement implique le formulaire auquel la méthode est attachée.

• **Méthodes table/triggers** : Un trigger est associé à une table. Vous ne pouvez pas appeler un trigger. 4D les exécute automatiquement à chaque opération élémentaire effectuée sur les enregistrements de la table (Chargement, Ajout, Suppression et Modification). Les triggers sont des méthodes pouvant prévenir toute action "illégale" opérée sur les enregistrements. Par exemple, dans un système de facturation, vous pouvez empêcher les utilisateurs d'ajouter des factures sans avoir spécifié le client concerné. Les triggers consituent un outil très puissant pour contrôler les opérations effectuées sur les tables ainsi que pour prévenir toute perte de données accidentelle. Vous pouvez écrire des triggers très simples ou très sophistiqués.

Pour plus d'informations sur les triggers, reportez-vous à la section Présentation des triggers.

• **Méthodes projet** : A la différence des précédents types de méthodes, les méthodes projet ne sont associées à aucun objet, formulaire ou table, elles peuvent être utilisées à n'importe quel endroit de votre base. Les méthodes projet sont réutilisables et disponibles à tout moment, pour toute autre méthode. Si vous devez répéter certaines tâches, vous n'avez pas besoin de réécrire plusieurs méthodes identiques dans chaque cas. Vous pouvez appeler une méthode projet partout où vous en avez besoin — depuis d'autres méthodes projet, objet ou formulaire. Lorsque vous appelez une méthode projet, elle se comporte comme si vous l'aviez écrite à l'endroit d'où vous l'appelez. Les méthodes projet utilisées par d'autres méthodes sont appelées des **sous-routines**. Une méthode projet qui retourne un résultat peut aussi être appelée une **fonction**.

Il existe une autre manière d'appeler les méthodes projet : il suffit de les associer à des commandes de menus. Lorsque vous associez une méthode projet à une commande de menu, la méthode est exécutée lorsque la commande du menu est sélectionnée par un utilisateur.

• **Méthodes base** : Tout comme les méthodes formulaire ou objet sont exécutées lorsqu'un événement se produit dans un formulaire, il existe des méthodes associées à la base entière, et qui sont exécutées lorsqu'un événement de session de travail se produit. Ce sont les **méthodes base**. Par exemple, à chaque fois vous ouvrez la base, vous pouvez vouloir initialiser des variables qui seront utilisées pendant toute la session de travail. Pour cela, vous pouvez écrire des instructions dans la méthode base Sur ouverture, exécutée automatiquement par 4D lorsque vous lancez la base.

Pour plus d'informations sur les méthodes base, reportez-vous à la section Présentation des méthodes base.

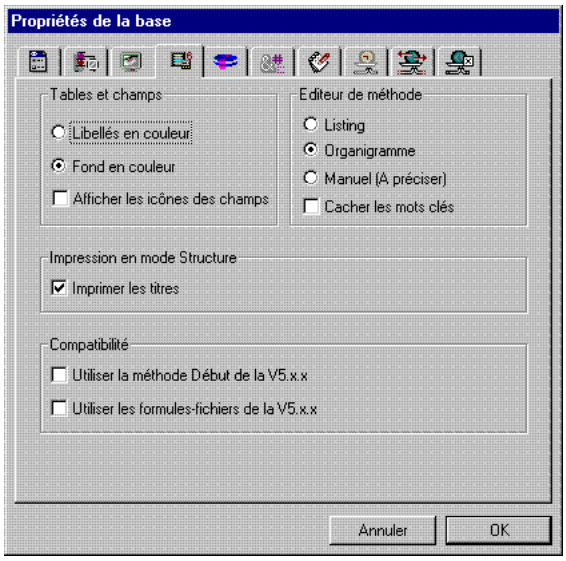
Notes de compatibilité
Il n'est pas nécessaire de lires ces notes si vous travaillez sur de nouvelles bases créées avec la version 6 de 4e Dimension.

(1) La version 6 de 4e Dimension a introduit de nombreux nouveaux objets ainsi que les événements formulaire (tels que Sur double clic, Sur gain focus, etc.) qui remplacent les cycles d'exécution des versions précédentes. Si vous avez converti une base version 5 en version 6, vos formulaires ont été convertis de manière à ce que leur comportement corresponde à ce que vous en attendez. Si vous souhaitez tirer parti des nouveaux événements formulaires avec des objets créés dans une version précédente de 4D, vous devrez sélectionner ces nouveaux événements dans les fenêtres des Propriétés des objets et des formulaires. Pour plus d'informations sur ce point, reportez-vous à la description de la commande Evenement formulaire et au manuel *Mode Structure* de 4e Dimension.

(2) Les triggers appartiennent à un nouveau type de méthodes introduit avec la version 6 de 4e Dimension. Dans les versions précédentes de 4e Dimension, les méthodes table (appelées alors formules-fichiers) n'étaient exécutées par 4D que lorsqu'un formulaire d'une table était utilisé pour la saisie, l'affichage ou l'impression. Elles étaient rarement utilisées. Il est à noter que les triggers sont exécutés à un niveau considérablement plus bas que ne l'étaient les formules-fichiers. Quelle que soit l'action réalisée sur un enregistrement (actions des utilisateurs — i.e. saisie — ou saisie par programmation — i.e. un appel à la commande STOCKER ENREGISTREMENT), le trigger d'une table sera appelé par 4D. Les triggers sont réellement différents des anciennes formules-fichiers. Si vous avez converti une base version 5 en version 6 de 4D et que vous souhaitez tirer avantage des triggers, vous devez désélectionner la propriété **Utiliser les anciennes formules-fichiers** dans la boîte de dialogue des Propriétés de la base, présentée ci-dessous.

(3) Les méthodes base sont un nouveau type de méthodes introduit avec la version 6 de 4e Dimension. Avec les versions précédentes de 4e Dimension, il n'existait qu'une méthode (appelée alors procédure) que 4D exécutait automatiquement au moment où vous ouvriez la base. Cette procédure devait être appelée STARTUP dans les versions de langue anglaise et DEBUT dans la version française afin d'être exécutée. Si vous avez converti une base version 5 en version 6 de 4D et que vous souhaitez tirer parti des capacités nouvelles de la Méthode base Sur ouverture, vous devez désélectionner la propriété **Utiliser la méthode Debut de la V5.x.x** dans la boîte de dialogue des Propriétés de la base, présentée ci-après. Cette propriété n'affecte que l'option Sur ouverture/Procédure début.

Si vous ne désélectionnez pas cette propriété et que vous ajoutez, par exemple, une Méthode base Sur fermeture, cette dernière sera quand même exécutée par 4D.



Un exemple de méthode projet

Toutes les méthodes sont fondamentalement identiques — elles débutent sur la première ligne et traitent chaque ligne d'instruction jusqu'à ce qu'elles atteignent la dernière ligne (c'est-à-dire qu'elles s'exécutent séquentiellement). Voici un exemple de méthode projet :

```
CHERCHER([Personnes])     ` Afficher l'éditeur de recherches
Si (OK=1)    ` L'utilisateur a cliqué sur OK (et non sur Annuler)
   Si (Enregistrements trouves ([Personnes])=0) ` Si aucun enregistrement n'est trouvé...
      AJOUTER ENREGISTREMENT([Personnes])
         ` Permettre à l'utilisateur d'ajouter un enregistrement
   Fin de si
Fin de si      ` Fin
```

Chaque ligne de l'exemple est une **ligne d'instruction**. Tout ce que vous écrivez dans le langage de 4D est appelé du code. Le code est exécuté — ce qui signifie que 4e Dimension effectue l'action spécifiée par le code.

Nous allons examiner très attentivement la première ligne ; nous irons plus rapidement par la suite :

CHERCHER([Personnes]) ` Afficher l'éditeur de recherches

Le premier mot de la ligne, CHERCHER, est une commande. Une commande est un élément du langage de 4e Dimension — elle effectue une action. Dans ce cas, CHERCHER affiche l'éditeur de recherches, toute comme le fait la commande **Recherche...** du menu **Sélection** en mode Utilisation.

Les parenthèses signifient qu'un **paramètre** est **passé** à la commande CHERCHER. Un paramètre (ou **argument**) est une valeur nécessaire à une commande pour remplir son rôle. Dans ce cas, [Personnes] est le nom d'une table. Les noms des tables sont toujours écrits entre crochets ([...]). Nous pouvons donc dire : « La table Personnes est un paramètre de la commande CHERCHER ». Une commande qui accepte plusieurs paramètres.

Enfin, il y a un **commentaire** à la fin de la ligne. Un commentaire vous informe (ainsi que toute personne qui examinera votre méthode) de ce qui se passe dans le code. Un commentaire est signalé par une apostrophe inversée (`). Dans une ligne, tout ce qui suit un signe de commentaire est ignoré lorsque le code est exécuté. Un commentaire peut être placé sur sa propre ligne, ou à la suite du code, comme dans l'exemple. N'hésitez pas à utiliser les commentaires dans vos méthodes ; la lecture et la compréhension de votre code sont facilitées, aussi bien pour vous-même que pour d'autres personnes.

Note : Un commentaire peut contenir jusqu'à 80 caractères.

La ligne suivante de notre exemple vérifie qu'aucun enregistrement n'a été trouvé :

Si (**Enregistrements trouves** ([Personnes])=0) ` Si aucun enregistrement n'est trouvé...

La ligne d'instruction Si est un **test conditionnel** — une instruction qui contrôle l'exécution au pas à pas de votre méthode. Le Si effectue un test, et si le test est VRAI, la ou les lignes suivantes sont exécutées. Enregistrements trouves est une fonction — c'est-à-dire une commande qui retourne une valeur. Ici, Enregistrements trouves retourne le nombre d'enregistrements de la sélection courante de la table passée en paramètre.

Note : Seule la première lettre du nom d'une fonction est en majuscule. C'est la convention d'écriture utilisée pour les fonctions de 4e Dimension.

Vous savez déjà ce qu'est la sélection courante : le groupe d'enregistrements avec lequel vous travaillez à un instant donné. Si le nombre d'enregistrements est égal à 0 (c'est-à-dire, si aucun enregistrement n'a été trouvé), la ligne de code suivante est exécutée :

```
AJOUTER ENREGISTREMENT([Personnes])
```

La commande AJOUTER ENREGISTREMENT affiche un formulaire pour permettre à l'utilisateur de créer un nouvel enregistrement. Notez que cette ligne comporte une indentation. 4e Dimension formate votre code automatiquement ; l'indentation vous permet d'identifier facilement ce qui est dépendant du test conditionnel (Si).

```
Fin de si      ` Fin
```

La ligne d'instruction Fin de si conclut la séquence d'instructions contrôlée par le test Si. Pour chaque ligne d'instruction de test conditionnel, votre code doit comporter une instruction correspondante indiquant au langage ou s'arrête le test. Nous vous conseillons de bien maîtriser les concepts évoqués dans ce chapitre. S'ils sont nouveaux pour vous, n'hésitez pas à relire les explications fournies jusqu'à ce qu'elles vous semblent claires.

Pour en savoir plus...

• Pour plus d'informations sur les méthodes objet et formulaire, reportez-vous à la description de la commande Evenement formulaire ainsi qu'au manuel *Mode Structure* de 4e Dimension.
• Pour travailler avec les triggers, reportez-vous à la section Présentation des triggers.
• Pour une description détaillée des méthodes projet, reportez-vous à la section Méthodes projet.
• Les méthodes base sont détaillées dans la section Présentation des méthodes base et ses sous-sections.

Référence

Conditions et boucles, Constantes, Nommer les objets du langage 4D, Opérateurs, Pointeurs, Présentation des méthodes base, Présentation des tableaux, Présentation des triggers, Types de données, Variables.

Les méthodes projet, comme leur nom l'indique, s'appliquent à la totalité de votre projet, c'est-à-dire votre base de données. Alors que les méthodes formulaire ou les méthodes objet par exemple sont associées à des formulaires ou des objets, une méthode projet est disponible partout — elle n'est associée à aucun élément particulier de la base. Une méthode projet peut tenir les rôles suivants, en fonction de la manière dont elle est exécutée et utilisée :

- Méthode de menu
- Sous-routine et fonction
- Méthode de gestion de process
- Méthode de gestion d'événements
- Méthode de gestion d'erreurs

Ces appellations ne qualifient pas la nature des méthodes (ce qu'elles sont), mais leur fonction (ce qu'elle font).

Une **méthode de menu** est une méthode projet appelée depuis une commande de menu personnalisé. Elle se comporte comme un agent de police chargé de la circulation, dirigeant les flux de votre application. Les méthodes de menu contrôlent, aiguillent lorsque c'est nécessaire, affichent les formulaires, génèrent des états ou encore gèrent votre base.

Le deuxième type de méthode projet peut être considéré comme une méthode asservie — d'autres méthodes lui demandent d'effectuer des tâches. Ce type de méthode est appelé **sous-routine**. Une sous-routine qui retourne une valeur est appelée une **fonction**.

Une **méthode de gestion de process** est une méthode projet appelée lorsqu'un process est démarré. Le process existera tant que la méthode sera en cours d'exécution. Pour plus d'informations sur les process, reportez-vous à la section Introduction aux process. A noter qu'une méthode de menu associée à une commande de menu pour laquelle la propriété **Démarrer un nouveau process** est sélectionnée, est aussi la méthode de gestion de process pour le process créé.

Une **méthode de gestion d'événements** est une méthode dédiée à la gestion des événements, qui s'exécute dans un process différent de celui de la méthode de gestion des process. Généralement, pour la gestion des événements, vous pouvez laisser 4D faire le gros du travail. Par exemple, lors de la saisie de données, 4D détecte les clics souris et les touches enfoncées, puis appelle les méthodes objet et formulaire correspondantes, vous permettant ainsi de prévoir dans ces méthodes les traitements appropriés aux événements. Dans d'autres circonstances, vous devez gérer directement les événements. Si, par exemple, vous exécutez une opération de longue durée (telle qu'une Boucle...Fin de boucle appliquée à chaque enregistrement), vous souhaitez pouvoir interrompre l'opération en tapant la touche **Esc**. Dans ce cas, une méthode de gestion d'événements est parfaitement adaptée. Pour plus d'informations, reportez-vous à la description de la commande APPELER SUR EVENEMENT.

Une **méthode de gestion d'erreurs** est une méthode projet d'interruption. Elle s'exécute à l'intérieur du process dans lequel elle a été installée à chaque fois qu'une erreur se produit. Pour plus d'informations, reportez-vous à la description de la commande APPELER SUR ERREUR.

Méthodes de menu

Une méthode de menu est appelée en mode Menus créés lorsque la commande de menu personnalisé à laquelle elle est associée est sélectionnée. Vous assignez la méthode à la commande de menu dans l'éditeur de menus de 4e Dimension. Lorsque l'utilisateur sélectionne la commande de menu, la méthode est exécutée. Ce fonctionnement est l'un des principaux aspects de la personnalisation d'une base de données. C'est en créant des menus qui appellent des méthodes de menu que vous personnalisez votre base. Reportez-vous au manuel *Mode Structure* de 4e Dimension pour plus d'informations sur l'éditeur de menus.

Les commandes de menus personnalisés peuvent déclencher une ou plusieurs actions. Par exemple, une commande de menu de saisie d'enregistrements peut appeler une méthode effectuant deux actions : afficher le formulaire entrée approprié et appeler la commande AJOUTER ENREGISTREMENT jusqu'à ce que l'utilisateur annule la saisie de nouveaux enregistrements.

L'automatisation de séquences d'actions est une possibilité très puissante du langage de programmation de 4D. A l'aide des menus créés, vous pouvez automatiser des séquences de tâches qui devraient sinon être exécutées manuellement dans le mode Utilisation. Avec les menus créés, vous permettez aux utilisateurs de naviguer plus facilement dans votre base.

Sous-routines

Lorsque vous avez écrit une méthode projet, elle devient partie intégrante du langage de la base dans laquelle elle a été créée. Vous pouvez alors l'appeler de la même manière que vous appelez les commandes intégrées de 4e Dimension. Une méthode projet utilisée de cette manière est appelée une sous-routine.

L'utilisation de sous-routines procure les avantages suivants :

- Réduction du code répétitif,
- Clarification des méthodes,
- Modification plus facile des méthodes,
- Création de code modulaire.

Imaginons par exemple que vous travaillez avec une base de clients. A mesure que vous construisez la base, vous vous apercevez que vous répétez souvent certaines tâches, telles que la recherche d'un client et la modification de son enregistrement. Le code nécessaire à l'accomplissement de cette opération pourrait être :

```
   ` Recherche d'un client
CHERCHER PAR EXEMPLE([Clients])
   ` Sélection du formulaire entrée
FORMULAIRE ENTREE([Clients];"Saisie de données")
   ` Modification de l'enregistrement du client
MODIFIER ENREGISTREMENT([Clients])
```

Si vous n'utilisez pas de sous-routines, vous devrez écrire ce code à chaque fois que vous voudrez modifier l'enregistrement d'un client. Si cette opération peut être réalisée dans dix endroits différents de votre base, vous devrez la réécrire dix fois. Grâce aux sous-routines, vous ne l'écrirez qu'une seule fois en tout. C'est le premier avantage des sous-routines : réduire la quantité de code à écrire.

Si le code ci-dessus était une méthode projet appelée MODIFIER CLIENT, vous l'exécuteriez simplement en inscrivant son nom dans une autre méthode. Par exemple, pour modifier l'enregistrement d'un client puis l'imprimer, vous n'auriez qu'à écrire :

```
MODIFIER CLIENT
IMPRIMER SELECTION([Clients])
```

Cette possibilité simplifie énormément vos méthodes. Dans l'exemple ci-dessus, il n'est pas nécessaire de savoir comment fonctionne la méthode MODIFIER CLIENT, mais uniquement ce qu'elle fait. C'est le deuxième avantage que vous pouvez tirer de l'utilisation de sous-routines : la clarification de votre code. Ainsi, ces méthodes deviennent en quelque sorte des extensions du langage de 4e Dimension.

Si vous devez modifier votre mode de recherche des clients, comme dans notre exemple, il vous suffit de modifier une seule méthode, et non dix. C'est un autre avantage des sous-routines : faciliter les modifications de votre code.

Avec les sous-routines, vous rendez votre code modulaire. Cela signifie simplement que vous dissociez votre code en modules (sous-routines), chacun d'entre eux effectuant une tâche logique. Examinez le code suivant, tiré d'une base de gestion de comptes chèques :

```
CHERCHER CHEQUES EMIS  ` Rechercher les chèques émis
RAPPROCHER COMPTE  ` Rapprocher le compte
IMPRIMER RELEVE  ` Imprimer un relevé
```

Même pour quelqu'un qui ne connaît pas la base, le code est clair. Il n'est pas nécessaire d'examiner chaque sous-routine. Elles peuvent contenir de nombreuses lignes d'instructions et effectuer des opérations complexes, mais l'important est ce qu'elles font.

Nous vous conseillons de découper votre code en tâches logiques, ou modules, à chaque fois que c'est possible.

Passer des paramètres aux méthodes

Vous aurez souvent besoin de fournir des valeurs à vos méthodes. Vous pouvez facilement effectuer cette opération grâce aux paramètres.

Les **paramètres** (ou arguments) sont des données dont les méthodes ont besoin pour s'exécuter — le terme "paramètres" ou "arguments" est utilisé indifféremment dans ce manuel. Des paramètres sont également passés aux commandes intégrées de 4e Dimension. Dans l'exemple ci-dessous, la chaîne "Bonjour" est un paramètre de la commande ALERTE :

```
ALERTE("Bonjour")
```

Les paramètres sont passés de la même manière aux méthodes. Par exemple, si la méthode FAIRE QUELQUE CHOSE accepte trois paramètres, l'appel à cette méthode pourrait être de la forme suivante :

```
FAIRE QUELQUE CHOSE(AvecCeci;EtCela;CommeCeci)
```

Les paramètres sont séparés par des points-virgules (;).

Dans la sous-routine (la méthode appelée), la valeur de chaque paramètre est automatiquement copiée séquentiellement dans des variables locales numérotées : $1, $2, $3, etc. La numérotation des variables locales représente l'ordre des paramètres.

Ces variables/paramètres locaux ne sont pas réellement les champs, variables ou expressions passées à la **méthode appelante** : elles contiennent simplement leurs valeurs.

A l'intérieur de la sous-routine, vous pouvez utiliser les paramètres $1, $2... de la même manière que vous utilisez les autres variables locales.

Puisque ces variables sont locales, elles ne sont définies qu'à l'intérieur de la sous-routine et sont effacées à la fin de son exécution. Pour cette raison, une sous-routine ne peut pas modifier au niveau de la méthode appelante la valeur réelle des champs ou des variables passé(e)s en paramètre. Par exemple :

```
    ` Voici une partie de la méthode MA METHODE
    ` ...
FAIRE QUELQUE CHOSE ([Personnes]Nom)  ` Admettons que [Personnes]Nom ="william"
ALERTE([Personnes]Nom)

    ` Voici le code de la méthode FAIRE QUELQUE CHOSE
$1:=Majusc($1)
ALERTE($1)
```

La boîte de dialogue d'alerte affichée par FAIRE QUELQUE CHOSE contiendra "WILLIAM" et celle affichée par MA METHODE contiendra "william". La méthode a modifié localement la valeur du paramètre $1, mais cela n'affecte pas la valeur du champ [Personnes]Nom passé en paramètre par la méthode MA METHODE.

Si vous voulez réellement que la méthode FAIRE QUELQUE CHOSE modifie la valeur du champ, deux solutions s'offrent à vous :

1. Plutôt que de passer le champ à la méthode, vous lui passez un pointeur :

```
    ` Voici une partie de la méthode MA METHODE
    ` ...
    ` Admettons que [Personnes]Nom = "william"
FAIRE QUELQUE CHOSE (->[Personnes]Nom)
ALERTE([Personnes]Nom)

    ` Voici le code de la méthode FAIRE QUELQUE CHOSE
$1->:=Majusc($1->)
ALERTE($1->)
```

Ici, le paramètre n'est pas le champ lui-même, mais un pointeur vers le champ. Ainsi, à l'intérieur de la méthode FAIRE QUELQUE CHOSE, $1 ne contient plus la valeur du champ mais un pointeur vers le champ. L'objet **référencé** par $1 ($1-> dans le code ci-dessus) est le champ lui-même. Par conséquent, la modification de l'objet référencé dépasse les limites de la sous-routine et le champ lui-même est affecté. Dans cet exemple, les deux boîtes de dialogue d'alerte afficheront "WILLIAM".

Pour plus d'informations sur les pointeurs, reportez-vous à la section Pointeurs.

2. Plutôt que la méthode FAIRE QUELQUE CHOSE “fasse quelque chose”, vous pouvez la réécrire de manière à ce qu'elle retourne une valeur :

```
    ` Voici une partie de la méthode MA METHODE
    ` ...
    ` Admettons que [Personnes]Nom est égal à "william"
[Personnes]Nom:=FAIRE QUELQUE CHOSE ([Personnes]Nom)
ALERTE([Personnes]Nom)

    ` Voici le code de la méthode FAIRE QUELQUE CHOSE
$0:=$1
ALERTE($0)
```

Une sous-routine retournant une valeur est appelée une fonction. Ce point est traité dans les paragraphes suivants.

Note de programmation avancée : Les paramètres à l'intérieur de la sous-routine sont accessibles par l'intermédiaire des variables locales $1, $2... De plus, des paramètres peuvent être optionnels et être référencés à l'aide de la syntaxe ${...}. Pour plus d'informations sur ce point, reportez-vous à la description de la commande Nombre de parametres.

Les fonctions, méthodes projet retournant une valeur

Les méthodes peuvent retourner des valeurs. Une méthode qui retourne une valeur est appelée une **fonction**.

Les commandes de 4D ou de plug-ins qui retournent une valeur sont également appelées fonctions.

Par exemple, la ligne d'instruction suivante utilise une fonction intégrée, Longueur, qui retourne la longueur d'une chaîne. La valeur retournée par Longueur est placée dans une variable appelée MaLongueur :

```
MaLongueur:=Longueur("Comment suis-je arrivé là ?")
```

Toute sous-routine peut retourner une valeur. La valeur à retourner est placée dans la variable locale $0.

Par exemple, la fonction suivante, appelée Majuscules4, retourne une chaîne dont les quatre premiers caractères ont été passés en majuscules :

```
$0:=Majusc(Sous chaine($1; 1; 4))+Sous chaine($1; 5)
```

Voici un exemple qui utilise la fonction Majuscules4 :

```
NouvellePhrase:=Majuscules4 ("Bien joué.")
```

Dans ce cas, la variable NouvellePhrase prend la valeur "BIEN joué."

Le **retour de fonction**, $0, est une variable locale à la sous-routine. Elle peut être utilisée en tant que telle à l'intérieur de la sous-routine. Par exemple, dans le cas de la méthode FAIRE QUELQUE CHOSE utilisée précédemment, $0 recevait d'abord la valeur de $1, puis était utilisée en tant que paramètre de la commande ALERTE. Dans une sous-méthode, vous pouvez utiliser $0 comme n'importe quelle autre variable locale. C'est 4D qui retourne sa valeur finale (sa valeur courante au moment où la sous-routine se termine) à la méthode appelée.

Méthode projet récursives

Des méthodes projet peuvent s'appeler les unes les autres, et en particulier :
• Une méthode A peut appeler une méthode B, qui appelle A, donc A appelle B de nouveau, etc.
• Une méthode peut s'appeler elle-même.

Cela s'appelle la **récursivité**. Le langage de 4D supporte pleinement la récursivité.

Examinons l'exemple suivant : vous diposez d'une table [Amis et relations] composée de l'ensemble de champs suivant (très simplifié) :
- [Amis et parents]Nom
- [Amis et parents]Enfant'Nom

Pour cet exemple, nous supposons que les valeurs des champs sont uniques (il n'existe pas deux personnes avec le même nom). A partir d'un nom, vous voulez écrire la phrase “Un de mes amis, Pierre, qui est le rejeton de Paul qui est le rejeton de Martine qui est le rejeton de Robert qui est le rejeton de Gertrude, fait cela pour gagner sa vie !” :

1. Vous pouvez procéder de la manière suivante :

```
$vsNom:=Demander("Saisissez le nom :";"Pierre")
Si (OK=1)
   CHERCHER([Amis et parents];[Amis et parents]Nom=$vsNom)
   Si (Enregistrements trouves([Amis et parents])>0)
      $vtHistoireComplète:="Un de mes amis, "+$vsNom
      Repeter
         CHERCHER([Amis et parents];[Amis et parents]Enfant'Nom=$vsNom)
         $vlResultRecherche:=Enregistrements trouves([Amis et parents])
         Si ($vlResultRecherche>0)
            $vtHistoireComplète:=$vtHistoireComplète+" qui est le rejeton de "+
                                                                  [Amis et parents]Nom
            $vsNom:=[Amis et parents]Nom
         Fin de si
      Jusque ($vlResultRecherche=0)
      $vtHistoireComplète:=$vtHistoireComplète+", fait cela pour gagner sa vie !"
      ALERTE($vtHistoireComplète)
   Fin de si
Fin de si
```

2. Vous pouvez également procéder ainsi :

```
$vsNom:=Demander("Saisissez le nom :";"Pierre")
Si (OK=1)
   CHERCHER([Amis et parents];[Amis et parents]Nom=$vsNom)
   Si (Enregistrements trouves([Amis et parents])>0)
      ALERTE("Un de mes amis, "+Généalogie de ($vsNom)+", fait cela pour
                                                                  gagner sa vie !")
   Fin de si
Fin de si
```

Vous utilisez la fonction récursive Généalogie de suivante :

```
` Méthode projet Généalogie de
` Généalogie de ( Chaîne ) -> Texte
` Généalogie de ( Nom ) -> Partie de la phrase

$0:=$1
CHERCHER([Amis et parents];[Amis et parents]Enfant'Nom=$1)
Si (Enregistrements trouves([Amis et parents])>0)
   $0:=$0+" qui est le rejeton de "+Généalogie de ([Amis et parents]Nom)
Fin de si
```

Vous notez que la méthode Généalogie de s'appelle elle-même.

La première manière de procéder utilise un **algorithme itératif**. La seconde manière utilise un **algorithme récursif**.

Lorsque vous implémentez du code pour traiter des cas comme celui décrit ci-dessus, vous aurez toujours le choix entre écrire des méthodes utilisant des algorithmes itératifs ou récursifs. Généralement, la récursivité permet d'écrire du code plus concis, lisible et plus facilement modifiable, mais utiliser la récursivité n'est absolument pas obligatoire.

Dans 4D, la récursivité est typiquement utilisée pour :
• Traiter les enregistrements de tables liées les unes aux autres de la même manière que décrit dans l'exemple ci-dessus.
• Naviguer parmi les documents et les dossiers de votre disque à l'aide des commandes LISTE DES DOSSIERS et LISTE DES DOCUMENTS. Un dossier peut contenir des dossiers et des documents, les sous-dossiers peuvent eux-mêmes contenir des dossiers et des documents, etc.

Important : Les appels récursifs doivent toujours se terminer à un moment donné. Dans l'exemple ci-dessus, la méthode Généalogie de cesse de s'appeler elle-même lorsque la recherche ne trouve plus d'enregistrement. Sans ce test conditionnel, la méthode s'appellerait indéfiniment et 4D pourrait au bout d'un certain temps retourner l'erreur “La pile est pleine” car le programme n'aurait plus assez de place pour "empiler" les appels (ainsi que les paramètres et les variables locales utilisés dans la méthode).

Référence

Conditions et boucles, Méthodes, Présentation des méthodes base.

3

BLOB

Definition

La version 6 de 4e Dimension a introduit le support des données de type BLOB (Binary Large OBjects).

Vous pouvez définir des champs de type BLOB et des variables de type BLOB :

• Pour créer un champ de type BLOB, sélectionnez BLOB dans la liste déroulante **Type de champ** dans la fenêtre des **Propriétés de champ**.
• Pour créer une variable de type BLOB, utilisez la directive de compilation C_BLOB. Vous pouvez créer des variables BLOB locales, process et interprocess.

Note : Les tableaux ne peuvent avoir le type BLOB.

Dans 4e Dimension, un BLOB est une série contiguë d'octets de longueur variable qui peut être traitée comme un seul objet ou dont les octets peuvent être adressés individuellement. Un BLOB peut être vide (longueur nulle) ou contenir jusqu'à 2147483647 octets (2 Go).

Les BLOBs et la mémoire

Lorsque vous travaillez avec un BLOB, il est stocké entièrement en mémoire. Si vous travaillez avec une variable de type BLOB, elle n'existe qu'en mémoire. Si vous travaillez avec un champ de type BLOB, il est chargé en mémoire à partie du disque, comme le reste de l'enregistrement auquel il appartient.

A l'instar des autres types de champs pouvant contenir une grande quantité de données (comme les champs de type Image et Sous-table), les champs de type BLOB ne sont pas dupliqués en mémoire lorsque vous modifiez un enregistrement. Par conséquent, les résultats renvoyés par Ancien et Modifie ne sont pas significatifs lorsque ces fonctions sont appliquées à des champs de type BLOB.

Afficher des BLOBs

Comme un BLOB peut contenir n'importe quel type de données, il n'existe pas de mode de représentation à l'écran par défaut des BLOBs. Si vous affichez un champ ou une variable de type BLOB dans un formulaire, l'objet sera toujours vide, quel que soit son contenu.

Champs de type BLOB

Vous pouvez utiliser des champs de type BLOB pour stocker tout type de données dont la taille est inférieure ou égale à 2 Giga-octets. Vous ne pouvez pas indexer un champ de type BLOB. Si vous voulez rechercher des enregistrements à partir d'une valeur contenue dans un champ BLOB, il sera nécessaire d'écrire une formule. N'utilisez pas de champs de type BLOB pour stocker des données sur lesquelles vous voulez effectuer des recherches rapides. Par exemple, ne stockez pas de mots-clés dans un champ de type BLOB, utilisez plutôt une sous-table que vous pourrez indexer.

Passage des paramètres, pointeurs et résultats de fonctions

Les BLOBs dans 4e Dimension peuvent être passés comme paramètres aux commandes 4D ou aux routines des Extensions 4D (plug-ins) qui attendent un paramètre de type BLOB. En revanche, les BLOBs ne peuvent pas être passés aux méthodes que vous créez. Un BLOB ne peut pas être retourné comme un résultat de fonction. Pour passer un BLOB à une de vos méthodes, il vous faut définir un pointeur vers le BLOB et passer le pointeur comme paramètre. Voici quelques exemples :

```
    ` Déclarer une variable de type BLOB
C_BLOB (touteVarBLOB)
    ` Le BLOB est passé comme paramètre à une commande 4D
FIXER TAILLE BLOB (touteVarBLOB;1024*1024)
    ` Le BLOB est passé comme paramètre à une routine externe
$CodeErr:= Faites_Quelque_chose_avec_ce_BLOB (touteVarBLOB)
    ` Un pointeur vers le BLOB est passé comme paramètre à une de vos méthodes
REMPLIR BLOB AVEC DES ZEROS (->touteVarBLOB )
    ` Déclarez une variable de type Pointeur
C_POINTEUR (aPointeur)
    ` Définir un pointeur vers le BLOB
aPointeur := ->touteVarBLOB
    ` Un pointeur vers le BLOB est passé comme paramètre à une de vos méthodes
REMPLIR BLOB AVEC DES ZEROS (aPointeur)
```

Note pour les développeurs de plug ins 4D : Un paramètre de type BLOB se déclare “&O” (la lettre “O” et non le chiffre “0”).

Affectation

Vous pouvez affecter la valeur d'un BLOB à d'autres BLOBs, comme dans l'exemple suivant :

```
    ` Déclarer deux variables de type BLOB
C_BLOB (vBlobA;vBlobB)
    ` Fixer la taille du premier BLOB à 10Ko
FIXER TAILLE BLOB (vBlobA;10*1024)
    ` Affectez le premier BLOB au second
vBlobB:=vBlobA
```

En revanche, il n'existe pas d'opérateur pouvant être utilisé avec des BLOBs ; il n'existe pas d'expression de type BLOB.

Adresser le contenu d'un BLOB

Chaque octet d'un BLOB peut être adressé individuellement, à l'aide des accolades {...}. Dans un BLOB, les octets sont numérotés de 0 à N-1, N étant la taille du BLOB. Voici un exemple :

```
    ` Déclarer une variable de type BLOB
C_BLOB (vBlob)
    ` Fixer la taille du BLOB à 256 octets
FIXER TAILLE BLOB (vBlob;256)
```

```
   ` La boucle suivante initialise les 256 octets du BLOB à zéro
Boucle (vOctet; 0;Taille BLOB (vBlob)-1)
   vBlob{vOctet}:=0
Fin de boucle
```

Comme pouvez adresser individuellement tous les octets d'un BLOB, vous pouvez littéralement stocker tout ce que vous voulez dans une variable ou un champ de type BLOB.

Routines 4e Dimension pour manipuler les BLOBs

4e Dimension fournit les routines suivantes pour travailler avec les BLOBs :

• FIXER TAILLE BLOB redimensionne la taille d'un champ ou d'une variable de type BLOB.
• Taille BLOB retourne la taille d'un champ ou d'une variable de type BLOB.
• DOCUMENT VERS BLOB et BLOB VERS DOCUMENT vous permettent de charger et d'écrire un document entier dans un champ ou une variable de type BLOB et inversement (en option, les data forks et les resource forks sur Macintosh).
• VARIABLE VERS BLOB et BLOB VERS VARIABLE, ainsi que LISTE VERS BLOB et BLOB vers liste vous permettent de stocker et de charger des variables 4D dans des BLOBs.
• COMPRESSER BLOB, DECOMPRESSER BLOB et LIRE PROPRIETES BLOB vous permettent de travailler avec des BLOBs compressés.
• Les commandes BLOB vers entier, BLOB vers entier long, BLOB vers reel, BLOB vers texte, ENTIER VERS BLOB, ENTIER LONG VERS BLOB, REEL VERS BLOB et TEXTE VERS BLOB vous permettent de manipuler et de structurer les données en provenance du disque, des ressources, du système d'exploitation, etc.
• SUPPRIMER DANS BLOB, INSERER DANS BLOB et COPIER BLOB permettent de gérer les gros volumes de données à l'intérieur des BLOBs.
• CRYPTER BLOB et DECRYPTER BLOB permettent de crypter et de décrypter des données dans vos bases 4D.

Ces commandes sont décrites dans ce chapitre. De plus, vous disposez des routines suivantes :

• C_BLOB déclare une variable de type BLOB (reportez-vous à la section Commandes du thème Compilateur).
• LIRE PRESSE PAPIERS et AJOUTER A PRESSE PAPIERS vous permettent de gérer tout type de données stockées dans le presse-papiers (référez-vous à la section Commandes du thème Presse-papiers).
• LIRE RESSOURCE et ECRIRE RESSOURCE vous permettent de gérer tout type de données stockées dans une ressource qui se trouve sur disque (référez-vous à la section Ressources).
• ENVOYER BLOB HTML vous permet d'envoyer tout type de données à un navigateur Web. Cette commande est incluse dans le thème Serveur Web.
• IMAGE VERS BLOB, BLOB VERS IMAGE et IMAGE VERS GIF vous permettent d'ouvrir et de convertir des images via des BLOBs. Ces commandes sont incluses dans le thème Images.
• GENERER CLES CRYPTAGE et GENERER DEMANDE CERTIFICAT permettent d'exploiter le protocole SSL (*Secured Socket Layer*) pour crypter des données ou les connexions Web. Ces commandes sont incluses dans le thème Protocole sécurisé.

FIXER TAILLE BLOB

BLOB

version 6.0

FIXER TAILLE BLOB (blob; taille{; remplisseur})

Paramètre	Type		Description
blob	BLOB	→	Champ ou variable de type BLOB
taille	Numérique	→	Nouvelle taille de BLOB
remplisseur	Numérique	→	Code ASCII du caractère de remplissage

Description

FIXER TAILLE BLOB redimensionne blob selon la valeur passée dans le paramètre taille.

Si vous souhaitez que les nouveaux octets réservés (s'il y en a) pour le BLOB soient initialisés avec une valeur particulière, passez cette valeur (comprise entre 0 et 255) dans le paramètre optionnel remplisseur.

Exemples

(1) Lorsque vous n'avez plus besoin d'un BLOB process ou interprocess, il est préférable de libérer la mémoire qu'il occupe. Pour cela, écrivez le code suivant :

```
⇒ FIXER TAILLE BLOB(vProcessBLOB;0)
⇒ FIXER TAILLE BLOB(◊vInterprocessBLOB;0)
```

(2) L'exemple suivant crée un BLOB de 16 Ko et remplit chaque octet avec la valeur 0xFF :

```
  C_BLOB(vxData)
⇒ FIXER TAILLE BLOB(vxData;16*1024;0xFF)
```

Référence

Taille BLOB.

Gestion des erreurs

Si vous ne pouvez pas redimensionner le BLOB parce qu'il n'y a pas assez de mémoire, l'erreur –108 est générée. Vous pouvez installer une méthode avec la commande APPELER SUR ERREUR pour interrompre la méthode lorsqu'une erreur survient.

Taille BLOB

BLOB

version 6.0

Taille BLOB (blob) → Entier long

Paramètre	Type		Description
blob	BLOB	→	Champ ou variable de type BLOB
Résultat	Entier long	←	Taille en octets du BLOB

Description

Taille BLOB retourne la taille de blob exprimée en octets.

Exemple

La ligne de code suivante ajoute 100 octets au BLOB monBlob :

```
⇒ FIXER TAILLE BLOB (Taille BLOB(monBlob)+100)
```

Référence

FIXER TAILLE BLOB.

COMPRESSER BLOB

BLOB

version 6.5.3 (Modifiée)

COMPRESSER BLOB (blob{; compression})

Paramètre	Type		Description
blob	BLOB	→	BLOB à compresser
compression	Numérique	→	Si ce paramètre est passé : 1= taux de compression maximum 2 = vitesse de compression maximum

Description

COMPRESSER BLOB compresse le BLOB blob à l'aide de l'algorithme de compression interne de 4e Dimension.

Le paramètre optionnel compression vous permet de fixer la façon dont le BLOB sera compressé :
• Si vous passez 1, le BLOB est compressé de manière aussi compacte que possible, au détriment de la vitesse à laquelle la compression et la décompression sont effectuées.
• Si vous passez 2, le BLOB est compressé de manière aussi rapide que possible (et sera décompressé aussi vite que possible) au détriment du taux de compression (une fois compressé, le BLOB prend plus de place).
• Si vous passez une autre valeur ou si vous omettez ce paramètre, le BLOB est compressé de manière aussi compacte que possible (méthode de compression 1).

4e Dimension fournit les constantes suivantes:

Constante	Type	Valeur
Méthode de compression compacte	Entier long	1
Méthode de compression rapide	Entier long	2

Note : La commande compresse uniquement les BLOBs de taille supérieure ou égale à 255 octets.

Après que cette commande ait été appelée, la variable système *OK* prend la valeur 1 si le BLOB a été correctement compressé.
Si la compression n'a pu être effectuée, *OK* prend la valeur 0. Dans ce cas, si l'erreur provient du fait que la taille du BLOB est inférieure à 255 octets ou que la mémoire disponible est insuffisante pour effectuer l'opération, aucune erreur n'est générée, la méthode poursuit son exécution.
En revanche, si l'erreur est causée par un problème plus important (le BLOB est endommagé), l'erreur -10600 est générée. Cette erreur, relativement rare, peut être interceptée à l'aide d'une méthode installée par la commande APPELER SUR ERREUR.

Lorsqu'un BLOB a été compressé, vous pouvez le décompresser à l'aide de la commande DECOMPRESSER BLOB.

Pour savoir si un BLOB a été compressé, utilisez la commande LIRE PROPRIETES BLOB.

ATTENTION : Un BLOB compressé est toujours un BLOB, rien ne vous empêche donc de modifier son contenu. Cependant, si vous le modifiez, la commande DECOMPRESSER BLOB ne pourra plus décompresser correctement le BLOB.

Exemples

(1) L'exemple suivant teste si le BLOB vxMonBlob est compressé et, sinon, le compresse :

```
LIRE PROPRIETES BLOB (vxMonBlob;$vlCompressé;$vlTailleDécompressée;
                                                    $vlTailleCourante)
Si ($vlCompressé=Non compressé)
⇒    COMPRESSER BLOB (vxMonBlob)
Fin de si
```

Notez que si vous appliquez COMPRESSER BLOB à un BLOB déjà compressé, la commande le détecte et ne fait rien.

(2) L'exemple suivant vous permet de sélectionner un document puis de le compresser :

```
$vhDocRef := Ouvrir document ("")
Si (OK=1)
   FERMER DOCUMENT ($vhDocRef)
   DOCUMENT VERS BLOB (Document;vxBlob)
   Si (OK=1)
⇒     COMPRESSER BLOB (vxBlob)
      Si (OK=1)
         BLOB VERS DOCUMENT (Document;vxBlob)
      Fin de si
   Fin de si
Fin de si
```

Référence

DECOMPRESSER BLOB, LIRE PROPRIETES BLOB.

Variables et ensembles système

La variable OK prend la valeur 1 si le BLOB a été correctement compressé, sinon elle prend la valeur 0.

DECOMPRESSER BLOB

BLOB

version 6.5.3 (Modifiée)

DECOMPRESSER BLOB (blob)

Paramètre	Type		Description
blob	BLOB	→	BLOB à décompresser

Description

DECOMPRESSER BLOB décompresse le BLOB blob préalablement compressé à l'aide de la commande COMPRESSER BLOB.

Après l'appel de la commande, la variable système *OK* prend la valeur 1 si le BLOB a été correctement décompressé.

Si la décompression n'a pas pu être effectuée, *OK* prend la valeur 0.
Dans ce cas, si l'erreur provient du fait que la mémoire disponible est insuffisante pour effectuer l'opération, aucune erreur n'est générée et la méthode poursuit son exécution. En revanche, si l'erreur est causée par un problème plus important (le BLOB n'avait pas été compressé ou le BLOB est endommagé), l'erreur -10600 est générée. Cette erreur peut être interceptée à l'aide d'une méthode installée par la commande APPELER SUR ERREUR.

De manière générale, il est préférable d'appeler la commande LIRE PROPRIETES BLOB pour savoir si le BLOB a été compressé avant d'exécuter DECOMPRESSER BLOB.

Exemples

(1) L'exemple suivant teste si le BLOB vxMonBlob est compressé et, si oui, le décompresse :

```
LIRE PROPRIETES BLOB (vxMonBlob;$vlCompressé;$vlTailleDécompressée;
                                                    $vlTailleCourante)
Si ($vlCompressé#Non compressé)
⇒    DECOMPRESSER BLOB (vxMonBlob)
Fin de si
```

(2) L'exemple suivant vous permet de sélectionner un document et puis de le décompresser s'il était compressé :

```
$vhDocRef := Ouvrir document ("")
Si (OK=1)
   FERMER DOCUMENT ($vhDocRef)
   DOCUMENT VERS BLOB (Document;vxBlob)
   Si (OK=1)
      LIRE PROPRIETES BLOB (vxBlob;$vlCompressé;$vlTailleDécompressée;
                                                          $vlTailleCourante)
      Si ($vlCompressé#Non compressé)
⇒        DECOMPRESSER BLOB (vxBlob)
         Si (OK=1)
            BLOB VERS DOCUMENT (Document;vxBlob)
         Fin de si
      Fin de si
   Fin de si
Fin de si
```

Référence

COMPRESSER BLOB, LIRE PROPRIETES BLOB.

Variables et ensembles système

La variable OK prend la valeur 1 si le BLOB a été correctement décompressé, sinon elle prend la valeur 0.

LIRE PROPRIETES BLOB

BLOB

version 6.0

LIRE PROPRIETES BLOB (blob; compressé{; tailleDécompressée{; tailleCourante}})

Paramètre	Type		Description
blob	BLOB	→	BLOB sur lequel vous voulez obtenir des informations
compressé	Numérique	←	0 = BLOB n'est pas compressé 1 = BLOB compressé méthode compacte 2 = BLOB compressé méthode rapide
tailleDécompressée	Numérique	←	Taille du BLOB décompressé en octets
tailleCourante	Numérique	←	Taille courante du BLOB en octets

Description

LIRE PROPRIETES BLOB retourne des informations sur le BLOB blob.

• Le paramètre compressé vous indique si le BLOB est compressé ou non et retourne une des valeurs suivantes :

Constante	Type	Valeur
Non compressé	Entier long	0
Méthode de compression compacte	Entier long	1
Méthode de compression rapide	Entier long	2

Note : Les constantes ci-dessus sont fournies par 4e Dimension.

• Quel que soit l'état de compression du BLOB, le paramètre tailleDécompressée retourne la taille du BLOB non compressé.
• Le paramètre tailleCourante retourne la taille courante du BLOB. Si le BLOB est compressé, tailleCourante sera inférieur à tailleDécompressée. Si le BLOB n'est pas compressé, tailleCourante sera égal à tailleDécompressée.

Exemples

(1) Référez-vous aux exemples des commandes COMPRESSER BLOB et DECOMPRESSER BLOB.

(2) Lorsqu'un BLOB est compressé, la méthode projet suivante vous permet de connaître le taux de place gagnée en compressant le BLOB :

```
` Méthode projet Place gagnée par compression
` Place gagnée par compression (Pointeur {; Pointeur } ) -> Entier long
` Place gagnée par compression ( -> BLOB {; -> octetsGagnés } ) -> Pourcentage

C_POINTEUR ($1;$2)
C_ENTIER LONG ($0;$vlCompressé;$vlTailleDécompressée;$vlTailleCourante)

⇒ LIRE PROPRIETES BLOB ($1->;$vlCompressé;$vlTailleDécompressée;$vlTailleCourante)
Si ($vlTailleDécompressée=0)
   $0:=0
   Si (Nombre de parametres>=2)
      $2->:=0
   Fin de si
Sinon
   $0:=100-(($vlTailleCourante/$vlTailleDécompressée)*100)
   Si (Nombre de parametres>=2)
      $2->:=$vlTailleDécompressée-$vlTailleCourante
   Fin de si
Fin de si
```

Lorsque cette méthode est placée dans votre application, vous pouvez écrire :

```
` ...
COMPRESSER BLOB (vxBlob)
$vlPourcent:=Place gagnée par compression  (->vxBlob;->vlTailleBlob)
ALERTE ("La compression permet de gagner "+Chaine (vlTailleBlob)
                              +" octets, donc "+Chaine($vlPourcent;"#0%")+" d'espace.")
```

Référence

COMPRESSER BLOB, DECOMPRESSER BLOB.

DOCUMENT VERS BLOB

BLOB

version 6.0

DOCUMENT VERS BLOB (document; blob{; *})

Paramètre	Type		Description
document	Alpha	→	Nom du document
blob	BLOB	→	Champ ou variable de type BLOB devant recevoir le document
		←	Contenu du document
*		→	Macintosh seulement : la resource fork est chargée si * est passé ; sinon, la data fork est chargée

Description

DOCUMENT VERS BLOB charge le contenu de document dans blob. Vous devez passer un nom de document valide, c'est-à-dire qui désigne à un document existant qui n'est pas déjà ouvert, sinon une erreur sera générée. Si vous voulez que l'utilisateur choisisse le document, utilisez la routine Ouvrir document et la variable système Document (cf. exemple ci-dessous).

Note pour les utilisateurs MacOS : les documents Macintosh peuvent être composés de deux éléments, la *resource fork* et la *data fork* ("partie des ressources" et "partie des données"). Par défaut, la commande DOCUMENT VERS BLOB charge la data fork du document. Si vous voulez charger la resource fork du document, passez le paramètre optionnel *.
Sous Windows, le paramètre * est ignoré. Notez que l'environnement 4D vous fournit l'équivalent des resource forks de MacOS sous Windows : par exemple, la data fork d'une base 4D est stockée dans un fichier comportant l'extension .4DB et la resource fork est stockée dans un fichier du même nom, mais comportant l'extension .RSR. Si vous développez une application 4D qui gère les data forks et les resource forks stockées dans des BLOBs, sous Windows, il vous suffit d'accéder au fichier correspondant à la "fork" avec laquelle vous voulez travailler.

Exemple

Notre exemple est une base qui vous permet de stocker et chercher rapidement des documents. Dans un formulaire entrée, vous créez un bouton qui vous permet de charger un document de votre choix dans un champ de type BLOB. La méthode pour ce bouton peut être la suivante :

```
$vhRefDoc:=Ouvrir document("") ` Sélectionner un document
Si (OK=1) ` Si un document a été choisi
   FERMER DOCUMENT($vhRefDoc) ` Nous voulons pas qu'il reste ouvert
      ` Charger le document
⇒     DOCUMENT VERS BLOB (Document;[VotreTable]VotreChampBLOB)
   Si (OK=0)
      ` Gérer l'erreur
   Fin de si
Fin de si
```

Référence

BLOB VERS DOCUMENT, Ouvrir document.

Variables système

La variable système OK prend la valeur 1 si le document est correctement lu. Sinon, elle prend la valeur 0 et une erreur est générée.

Gestion des erreurs

• Si vous essayez de charger dans un BLOB un document qui n'existe pas ou qui est déjà ouvert par un(e) autre process ou application, une des Erreurs du gestionnaire de fichiers du système sera générée.

• Une erreur d'Entrée/Sortie peut survenir si le document est verrouillé, s'il se trouve sur un volume verrouillé ou si un problème survient pendant la lecture du document.

• S'il n'y a pas assez de mémoire pour charger le document, une erreur -108 est générée.

Dans tous les cas, vous pouvez gérer les erreurs en utilisant la commande APPELER SUR ERREUR.

BLOB VERS DOCUMENT

BLOB

version 6.0

BLOB VERS DOCUMENT (document; blob{; *})

Paramètre	Type		Description
document	Alpha	→	Nom du document
blob	BLOB	→	Nouveau contenu du document
*	*	→	Macintosh seulement : si * est passé, la resource fork est écrite ; sinon, la data fork est écrite

Description

BLOB VERS DOCUMENT écrit le contenu de document en utilisant les données stockées dans blob.
Vous pouvez passer dans document le nom d'un document existant ou non. Si le document n'existe pas, la commande le crée. Si vous passez le nom d'un document existant, assurez-vous qu'il n'est pas déjà ouvert, sinon une erreur est générée. Si vous voulez que l'utilisateur choisisse le document, appelez les routines Ouvrir document ou Creer document et utilisez la variable système Document (cf. exemple ci-dessous).

Note pour les utilisateurs MacOS : les documents Macintosh peuvent être composés de deux éléments, la *resource fork* et la *data fork* ("partie des ressources" et "partie des données"). Par défaut, la commande BLOB VERS DOCUMENT réécrit la data fork du document. Si vous voulez réécrire la resource fork du document, passez le paramètre optionnel *.
Sous Windows, le paramètre * est ignoré. Notez que l'environnement 4D vous fournit l'équivalent des resource forks de MacOS sous Windows : par exemple, la data fork d'une base 4D est stockée dans un fichier comportant l'extension .4DB et la resource fork est stockée dans un fichier du même nom, mais comportant l'extension .RSR. Si vous développez une application 4D qui gère les data forks et les resource forks stockées dans des BLOBs, sous Windows, il vous suffit d'accéder au fichier correspondant à la "fork" avec laquelle vous voulez travailler.

Exemple

Notre exemple est une base qui permet de stocker et de rechercher rapidement des documents. Dans un formulaire entrée, vous créez un bouton vous permettant de sauvegarder un document de votre choix qui contient des données provenant d'un champ de type BLOB. La méthode de ce bouton peut être la suivante :

```
$vhRefDoc:=Ouvrir document("") ` Sélectionner un document
Si (OK=1) ` Si un document a été choisi
   FERMER DOCUMENT($vhRefDoc) ` Nous ne voulons pas qu'il reste ouvert
      ` Ecrire le contenu du document
⇒  BLOB VERS DOCUMENT(Document;[VotreTable]VotreChampBLOB)
   Si (OK=0)
      ` Gérer l'erreur
   Fin de si
Fin de si
```

Référence

Creer document, DOCUMENT VERS BLOB, Ouvrir document.

Variables système

La variable système OK prend la valeur à 1 si le document est correctement écrit. Sinon, elle prend la valeur 0 et une erreur est générée.

Gestion des erreurs

• Si vous essayez de réécrire un document qui est déjà ouvert par un autre process ou une autre application, une des Erreurs du gestionnaire de fichiers du système sera générée.

• L'espace sur disque peut être insuffisant pour l'écriture du contenu du document.

• Une erreur d'Entrée/Sortie peut survenir si le document est verrouillé, s'il se trouve sur un volume verrouillé ou si un problème survient lors de l'écriture du document.

Dans tous les cas, vous pouvez gérer les erreurs en utilisant la commande APPELER SUR ERREUR.

VARIABLE VERS BLOB

BLOB

version 6.0

VARIABLE VERS BLOB (variable; blob{; offset | *})

Paramètre	Type		Description
variable	Variable	→	Variable à stocker dans le BLOB
blob	BLOB	→	BLOB devant recevoir la variable
offset | *	Variable | *	→	Offset de la variable (en octets) dans BLOB ou * pour ajouter la variable à la fin du BLOB
		←	Nouvel offset après écriture si * omis

Description

VARIABLE VERS BLOB stocke la variable variable dans le BLOB blob.

Si vous passez le paramètre optionnel *, la variable est ajoutée à la fin de blob et la taille du BLOB est redimensionnée en conséquence. A l'aide du paramètre optionnel *, vous pouvez stocker les unes derrière les autres autant de variables ou de listes (cf. les autres commandes BLOB) que vous voulez dans un BLOB, la seule limite étant celle de la mémoire disponible.

Si vous ne passez pas le paramètre optionnel * ni de variable dans le paramètre offset, variable est stockée à partir du début du BLOB en écrasant son contenu précédent. La taille du BLOB est redimensionnée en conséquence.

Si vous passez la variable offset en paramètre, la variable est écrite dans le BLOB à l'offset (à partir de zéro) spécifié par offset. Quel que soit l'endroit où vous placez la variable, la taille du BLOB sera augmentée si nécessaire en fonction de l'emplacement que vous avez défini (ainsi que de la taille de la variable). Les octets redéfinis (autres que ceux que vous venez d'écrire) sont initialisés à la valeur zéro.

Après l'exécution de la commande, la variable du paramètre offset est incrémentée du nombre d'octets ayant été écrits. Vous pouvez par conséquent réutiliser la même variable avec une autre commande d'écriture de BLOB afin de placer une autre variable ou liste juste après celle que vous venez d'écrire.

VARIABLE VERS BLOB accepte tous les types de variables (y compris d'autres BLOBs), à l'exception des types suivants :
- Pointeurs
- Tableaux de pointeurs
- Tableaux à deux dimensions

Notez cependant que si vous stockez une variable de type Entier long qui est une référence à une liste hiérarchique (ListRef), VARIABLE VERS BLOB stockera la variable Entier long, pas la liste. Pour stocker et récupérer des listes hiérarchiques dans un BLOB, utilisez les commandes LISTE VERS BLOB et BLOB vers liste.

ATTENTION : Si vous utilisez un BLOB pour stocker les variables, utilisez par la suite la commande BLOB VERS VARIABLE pour récupérer le contenu du BLOB car les variables sont stockées dans les BLOBs avec un format interne à 4D.

La variable OK prend la valeur 1 si la variable a été correctement stockée. Si l'opération n'a pas pu être effectuée à cause d'un manque de mémoire, la variable OK prend la valeur 0.

Note sur l'indépendance de plate-forme : VARIABLE VERS BLOB et BLOB VERS VARIABLE utilisent un format interne à 4D pour gérer les variables stockées dans les BLOBs. Vous n'avez donc pas besoin de vous préoccuper de la conversion des octets ("byte swapping") entre les différentes plates-formes lors de l'utilisation de ces deux commandes. Un BLOB créé sous Windows à l'aide de ces deux commandes peut être réutilisé sans la moindre manipulation sous MacOS et vice-versa.

Exemples

(1) Les méthodes projet suivantes vous permettent de stocker et de récupérer rapidement des variables dans les documents sur disque :

```
   ` Méthode projet STOCKER VARIABLES
   ` STOCKER VARIABLES ( Alpha ; Pointeur )
   ` STOCKER VARIABLES ( Document ; -> Tableau )
C_ALPHA (255;$1)
C_POINTEUR ($2)
C_BLOB ($vxDonnéesTableau)
⇒ VARIABLE VERS BLOB ($2->;$vxDonnéesTableau)  ` Stocker le tableau dans le BLOB
COMPRESSER BLOB ($vxDonnéesTableau)  ` Compresser le BLOB
BLOB VERS DOCUMENT ($1;$vxDonnéesTableau)  ` Enregistrer le BLOB sur disque

   ` Méthode projet CHARGER VARIABLES
   ` CHARGER VARIABLES ( Alpha ; Pointeur )
   ` CHARGER VARIABLES ( Document ; -> Tableau )
C_ALPHA (255;$1)
C_POINTEUR ($2)
C_BLOB ($vxDonnéesTableau)
DOCUMENT VERS BLOB ($1;$vxDonnéesTableau)  ` Charger le BLOB du disque
DECOMPRESSER BLOB ($vxDonnéesTableau)  ` Décompresser le BLOB
BLOB VERS VARIABLE ($vxDonnéesTableau;$2->)  ` Récupérer le tableau du BLOB
```

Lorsque ces méthodes ont été ajoutées à votre application, vous pouvez écrire :

```
TABLEAU ALPHA (...;taToutTableau;...)
   ` ...
STOCKER VARIABLES ( $vaNomDoc;->taToutTableau)
   ` ...
CHARGER VARIABLES ( $vaNomDoc;->taToutTableau)
```

(2) Les deux méthodes projet suivantes vous permettent de stocker et de récupérer des variables dans un BLOB :

```
   ` Méthode projet STOCKER VARIABLES DANS BLOB
   ` STOCKER VARIABLES DANS BLOB ( Pointeur { ; Pointeur ... { ;  Pointeur } } )
   ` STOCKER VARIABLES DANS BLOB ( BLOB { ; Var1 ... { ; Var2 } } )
C_POINTEUR ($1)
C_ENTIER LONG ($vlParam)

FIXER TAILLE BLOB ($1->;0)
Boucle ($vlParam;2;Nombre de parametres)
⇒     VARIABLE VERS BLOB (${$vlParam}->;$1->;*)
Fin de boucle


   ` Méthode projet RECUPERER VARIABLES DANS BLOB
   ` RECUPERER VARIABLES DANS BLOB ( Pointeur { ; Pointeur ... { ;  Pointeur } } )
   ` RECUPERER VARIABLES DANS BLOB ( BLOB { ; Var1 ... { ; Var2 } } )
C_POINTEUR ($1)
C_ENTIER LONG ($vlParam;$vlOffset)

$vlOffset:=0
Boucle ($vlParam;2;Nombre de parametres)
   BLOB VERS VARIABLE ($1->;${$vlParam}->;$vlOffset)
Fin de boucle
```

Lorsque ces méthodes ont été ajoutées à votre application, vous pouvez écrire :

```
STOCKER VARIABLES DANS BLOB ( ->vxBLOB;->vgImage;->taTableau1;->taTableau2)
   ` ...
RECUPERER VARIABLES DANS BLOB ( ->vxBLOB;->vgImage;->taTableau1;->taTableau2)
```

Référence

BLOB vers liste, BLOB VERS VARIABLE, LISTE VERS BLOB.

Variables et ensembles système

La variable OK prend la valeur 1 si la variable a été correctement stockée, sinon elle prend la valeur 0.

BLOB VERS VARIABLE

BLOB

version 6.0

BLOB VERS VARIABLE (blob; variable{; offset})

Paramètre	Type		Description
blob	BLOB	→	BLOB contenant une ou plusieurs variable(s) 4D
variable	Variable	←	Variable à écrire avec le contenu de BLOB
offset	Numérique	→	Position de la variable dans BLOB
		←	Position de la variable suivante dans BLOB

Description

BLOB VERS VARIABLE réécrit la variable variable avec les données stockées dans le BLOB blob à l'offset d'octet (à partir de zéro) spécifié par offset.

Les données dans le BLOB doivent être compatibles avec la variable de destination : vous utiliserez généralement des BLOBs que vous avez précédemment remplis à l'aide de VARIABLE VERS BLOB.

Si vous ne spécifiez pas le paramètre offset, les données de la variable sont lues à partir du début du BLOB. Si le BLOB contient plusieurs variables, vous devez passer le paramètre offset ainsi qu'une variable numérique. Avant d'appeler la commande, définissez cette variable numérique avec l'offset correspondant. Après l'appel, la même variable numérique retourne l'offset de la variable suivante stockée dans le BLOB.

La variable OK prend la valeur 1 si l'opération s'est correctement déroulée. Si l'opération n'a pas pu être effectuée, par exemple à cause d'un manque de mémoire, la variable OK prend la valeur 0.

Note sur l'indépendance de plate-forme : BLOB VERS VARIABLE et VARIABLE VERS BLOB utilisent un format interne à 4D pour gérer les variables stockées dans les BLOBs. Vous n'avez donc pas besoin de vous préoccuper de la conversion des octets ("byte swapping") entre les différentes plates-formes lors de l'utilisation de ces deux commandes. Un BLOB créé sous Windows à l'aide de ces deux commandes peut être réutilisé sans la moindre manipulation sous MacOS et vice-versa.

Exemple

Référez-vous aux exemples de VARIABLE VERS BLOB.

Référence

VARIABLE VERS BLOB.

Variables et ensembles système

La variable OK prend la valeur 1 si la variable a été correctement réécrite, sinon elle prend la valeur 0.

LISTE VERS BLOB

BLOB

version 6.0

LISTE VERS BLOB (liste; blob{; *})

Paramètre	Type		Description
liste	RefList	→	Liste hiérarchique à stocker dans le BLOB
blob	BLOB	→	BLOB devant recevoir la liste hiérarchique
*	*	→	Ajouter la liste à la fin du BLOB

Description

La commande LISTE VERS BLOB stocke la liste hiérarchique liste dans le BLOB blob.

Si vous passez le paramètre optionnel *, la liste hiérarchique est ajoutée à la fin du BLOB et la taille de blob est modifiée en conséquence. Ainsi, à l'aide du paramètre optionnel *, vous pouvez stocker les unes derrière les autres autant de variables ou de listes (référez-vous aux autres commandes sur les BLOBs) que vous voulez dans un BLOB, la seule limite étant celle de la mémoire disponible.

Si vous ne passez pas le paramètre optionnel *, la liste hiérarchique est stockée au début de blob en remplaçant son contenu précédent, et la taille du BLOB est modifiée en conséquence.

Quel que soit l'endroit où vous placez la liste, la taille du BLOB sera augmentée si nécessaire en fonction de l'emplacement que vous avez défini (plus jusqu'à la taille de la liste le cas échéant). Les octets redéfinis (autres que ceux que vous venez d'écrire) sont initialisés à la valeur zéro.

ATTENTION : Si vous utilisez un BLOB pour stocker des listes, appelez ensuite la commande BLOB vers liste pour relire le contenu du BLOB car les listes sont stockées dans les BLOBs avec un format interne 4D.

Après l'exécution de la commande, la variable OK prend la valeur 1 si la liste hiérarchique a été correctement stockée. Si l'opération n'a pas pu être effectuée car, par exemple, il n'y avait pas assez de mémoire disponible, la variable OK prend la valeur 0.

Note pour l'indépendance de plate-forme : LISTE VERS BLOB et BLOB vers liste utilisent un format interne 4D pour gérer les listes stockées dans des BLOBs. L'avantage est que vous n'avez pas besoin de vous soucier de la conversion des octets ("byte swapping") entre les plates-formes lorsque vous utilisez ces deux commandes. Autrement dit, avec ces commandes, un BLOB créé sous Windows peut être réutilisé sous MacOS et vice-versa.

Exemple

Reportez-vous à l'exemple de la fonction BLOB vers liste.

Référence

BLOB vers liste, BLOB VERS VARIABLE, VARIABLE VERS BLOB.

Variables et ensembles système

La variable OK prend la valeur 1 si la liste a été correctement écrite, sinon elle prend la valeur 0.

BLOB vers liste

BLOB

version 6.0

BLOB vers liste (blob{; offset}) → RefList

Paramètre	Type		Description
blob	BLOB	→	BLOB contenant la liste hiérarchique
offset	Numérique	→	Offset (en octets) dans le BLOB
		←	Nouvel offset après la lecture
Résultat	RefList	←	Référence de la liste nouvellement créée

Description

BLOB vers liste crée une nouvelle liste hiérarchique avec les données stockées dans le BLOB blob à l'offset d'octet (à partir de zéro) spécifié par offset et retourne un numéro de référence de liste hiérarchique pour cette nouvelle liste.

Les données présentes dans le BLOB doivent être compatibles avec la commande : généralement, vous utilisez des BLOBs préalablement remplis avec la commande LISTE VERS BLOB.

Si vous ne passez pas le paramètre optionnel offset, les valeurs de la liste sont lues à partir du début du BLOB. Si vous gérez un BLOB dans lequel plusieurs variables ou listes ont été stockées, vous devez passer le paramètre offset ainsi qu'une variable numérique. Avant l'appel, fixez cette variable numérique à l'offset désiré. Après l'appel, cette même variable numérique retourne l'offset de la variable suivante stockée dans le BLOB.

Après l'appel, la variable OK prend la valeur 1 si la liste hiérarchique a été correctement créée. Si l'opération ne peut pas être effectuée à cause, par exemple, d'un manque de mémoire, la variable OK prend la valeur 0.

Note pour l'indépendance de plate-forme : BLOB vers liste et LISTE VERS BLOB utilisent un format interne 4D pour gérer les listes stockées dans des BLOBs. L'avantage est que vous n'avez pas besoin de vous soucier de la conversion des octets ("byte swapping") entre les plates-formes lorsque vous utilisez ces deux commandes. Autrement dit, avec ces commandes, un BLOB créé sous Windows peut être réutilisé sous MacOS et vice-versa.

Exemple

Dans l'exemple suivant, la méthode d'un formulaire entrée extrait une liste d'un champ BLOB avant que le formulaire ne s'affiche puis le stocke dans le champ BLOB lorsque la saisie est validée :

```
      ` Méthode du formulaire [Choses à Faire];"Entrée"

   Au cas ou

      : (Evenement formulaire=Sur chargement)
⇒        hListe:=BLOB vers liste([Choses à Faire]Idées)
         Si (OK=0)
            hListe:=Nouvelle liste
         Fin de si

      : (Evenement formulaire=Sur libération)
         SUPPRIMER LISTE(hListe;*)

      : (bValider=1)
         LISTE VERS BLOB(hListe;[Choses à Faire]Idées)

   Fin de cas
```

Référence

LISTE VERS BLOB.

Variables et ensembles système

La variable OK prend la valeur 1 si la liste a été correctement créée, sinon elle prend la valeur 0.

ENTIER VERS BLOB

BLOB

version 6.0

ENTIER VERS BLOB (entier; blob; ordreOctet{; offset | *})

Paramètre	Type		Description
entier	Numérique	→	Valeur entière à écrire dans le BLOB
blob	BLOB	→	BLOB devant recevoir la valeur entière
ordreOctet	Numérique	→	0 Ordre des octets en mode natif 1 Ordre des octets Macintosh 2 Ordre des octets PC
offset | *	Variable | *	→	Offset (en octets) de l'entier dans le BLOB ou * pour ajouter la valeur à la fin du BLOB
		←	Nouvel offset après écriture si * omis

Description

ENTIER VERS BLOB écrit la valeur entière (2 octets) entier dans le BLOB blob.

Le paramètre ordreOctet fixe l'ordre des octets ("byte ordering") de la valeur entière à écrire. Vous pouvez passer une des constantes fournies par 4e Dimension :

Constante	Type	Valeur
Ordre octets natif	Entier long	0
Ordre octets Macintosh	Entier long	1
Ordre octets PC	Entier long	2

Note sur l'indépendance de plate-forme : Si vous échangez des BLOBs entre les plates-formes Macintosh et PC, il vous incombe de traiter les conversions d'octets ("byte swapping") lorsque vous utilisez cette commande.

Si vous passez le paramètre optionnel *, la valeur entière sur 2 octets est ajoutée à la fin du BLOB et sa taille est modifiée en conséquence. Ainsi, à l'aide du paramètre optionnel *, vous pouvez stocker les unes derrière les autres autant de valeurs de type Entier, Entier long, Numérique ou Texte (référez-vous aux autres commandes sur les BLOBs) que vous voulez dans un BLOB, la seule limite étant celle de la mémoire disponible.

Si vous ne passez pas le paramètre optionnel * ni de variable dans le paramètre offset, la valeur entière est stockée au début de blob en remplaçant son contenu précédent, et la taille du BLOB est modifiée en conséquence.

Si vous passez une variable dans le paramètre offset, la valeur entière est écrite à partir de l'offset offset, exprimé en octets (à partir de zéro), du BLOB. Quel que soit l'endroit où vous placez l'entier, la taille du BLOB sera augmentée si nécessaire en fonction de l'emplacement que vous avez défini (plus jusqu'à 2 octets le cas échéant). Les octets redéfinis (autres que ceux que vous venez d'écrire) sont initialisés à la valeur zéro.

Après l'exécution de la commande, la variable du paramètre offset est incrémentée du nombre d'octets ayant été écrits. Vous pouvez par conséquent réutiliser la même variable avec une autre commande d'écriture de BLOB afin de placer une autre valeur juste après celle que vous venez d'écrire.

Exemples

(1) Après l'exécution de ce code :

⇒ **ENTIER VERS BLOB** (0x0206;vxBlob;Ordre octets natif)

• La taille de vxBlob est 2 octets
• Sur Macintosh vxBLOB{0} = $02 et vxBLOB{1} = $06
• Sur PC vxBLOB{0} = $06 et vxBLOB{1} = $02

(2) Après l'exécution de ce code :

⇒ **ENTIER VERS BLOB** (0x0206;vxBlob;Ordre octets Macintosh)

• La taille de vxBlob est 2 octets
• Sur toutes les plates-formes vxBLOB{0} = $02 et vxBLOB{1} = $06

(3) Après l'exécution de ce code :

⇒ **ENTIER VERS BLOB** (0x0206;vxBlob;Ordre octets PC)

• La taille de vxBlob est 2 octets
• Sur toutes les plates-formes vxBLOB{0} = $06 et vxBLOB{1} = $02

(4) Après l'exécution de ce code:

FIXER TAILLE BLOB (vxBlob;100)
⇒ **ENTIER VERS BLOB** (0x0206;vxBlob;Ordre octets PC;*)

• La taille de vxBlob est 102 octets
• Sur toutes les plates-formes vxBLOB{100} = $06 et vxBLOB{101} = $02
• Les autres octets du BLOB restent inchangés

(5) Après l'exécution de ce code :

FIXER TAILLE BLOB (vxBlob;100)
vlOffset:=50
⇒ **ENTIER VERS BLOB** (518;vxBlob;Ordre octets Macintosh;vlOffset)

- La taille de vxBlob est 100 octets
- Sur toutes les plates-formes vxBLOB{50} = $02 et vxBLOB{51} = $06
- Les autres octets du BLOB restent inchangés
- La variable vlOffset est incrémentée de 2 (et est alors égale à 52)

Référence

BLOB vers entier, BLOB vers entier long, BLOB vers reel, BLOB vers texte, ENTIER LONG VERS BLOB, REEL VERS BLOB, TEXTE VERS BLOB.

ENTIER LONG VERS BLOB

BLOB

version 6.0

ENTIER LONG VERS BLOB (entierLong; blob; ordreOctets{; offset | *})

Paramètre	Type		Description
entierLong	Numérique	→	Valeur de type Entier long à écrire dans BLOB
blob	BLOB	→	BLOB devant recevoir l'entier long
ordreOctets	Numérique	→	0 Ordre d'octets natif 1 Ordre d'octets Macintosh 2 Ordre d'octets PC
offset | *	Variable | *	→	Offset (en octets) dans le BLOB ou * pour ajouter la valeur à la fin du BLOB
		←	Nouvel offset après l'écriture si * omis

Description

La commande ENTIER LONG VERS BLOB écrit la valeur de type Entier long (4 octets) entierLong dans le BLOB blob.

Le paramètre ordreOctet fixe l'ordre des octets ("byte ordering") de la valeur Entier long à écrire. Vous passez une des constantes fournies par 4e Dimension :

Constante	Type	Valeur
Ordre octets natif	Entier long	0
Ordre octets Macintosh	Entier long	1
Ordre octets PC	Entier long	2

Note sur l'indépendance de plate-forme : Si vous échangez des BLOBs entre les plates-formes Macintosh et PC, il vous incombe de traiter les conversions d'octets ("byte swapping") lorsque vous utilisez cette fonction.

Si vous passez le paramètre optionnel *, l'entier long sur 4 octets est ajouté à la fin du BLOB et la taille de blob est modifiée en conséquence. Ainsi, à l'aide du paramètre optionnel *, vous pouvez stocker les unes derrière les autres autant de valeurs de type Entier, Entier long, Numérique ou Texte (référez-vous aux autres commandes sur les BLOBs) que vous voulez dans un BLOB, la seule limite étant celle de la mémoire disponible.

Si vous ne passez pas le paramètre optionnel * ni de variable dans le paramètre offset, l'entier long est stocké au début de blob en remplaçant son contenu précédent, et la taille du BLOB est modifiée en conséquence.

Si vous passez une variable dans le paramètre offset, l'entier long est écrit à partir de l'offset offset, exprimé en octets (à partir de zéro), du BLOB. Quel que soit l'endroit où vous placez l'entier long, la taille du BLOB sera augmentée si nécessaire en fonction de l'emplacement que vous avez défini (plus jusqu'à 4 octets le cas échéant). Les octets redéfinis (autres que ceux que vous venez d'écrire) sont initialisés à la valeur zéro.

Après l'exécution de la commande, la variable du paramètre offset est incrémentée du nombre d'octets ayant été écrits. Vous pouvez par conséquent réutiliser la même variable avec une autre commande d'écriture de BLOB afin de placer une autre valeur juste après celle que vous venez d'écrire.

Exemples

(1) Après l'exécution de ce code :

⇒ **ENTIER LONG VERS BLOB** (0x01020304;vxBlob;Ordre octets natif)

• La taille de vxBlob est 4 octets
• SurMacintosh vxBLOB{0}=$01, vxBLOB{1}=$02, vxBLOB{2}=$03, vxBLOB{3}=$04
• Sur PC vxBLOB{0}=$04, vxBLOB{1}=$03, vxBLOB{2}=$02, vxBLOB{3}=$01

(2) Après l'exécution de ce code :

⇒ **ENTIER LONG VERS BLOB** (0x01020304;vxBlob;Ordre octets Macintosh)

• La taille de vxBlob est 4 octets
• Sur toutes les plates-formes vxBLOB{0}=$01, vxBLOB{1}=$02, vxBLOB{2}=$03, vxBLOB{3}=$04

(3) Après l'exécution de ce code :

⇒ **ENTIER LONG VERS BLOB** (0x01020304;vxBlob;Ordre octets PC)

• La taille de vxBlob est 4 octets
• Sur toutes les plates-formes vxBLOB{0}=$04, vxBLOB{1}=$03, vxBLOB{2}=$02, vxBLOB{3}=$01

(4) Après l'exécution de ce code :

FIXER TAILLE BLOB (vxBlob;100)
⇒ **ENTIER LONG VERS BLOB** (0x01020304;vxBlob;Ordre octets PC)

• La taille de vxBlob est 104 octets
• Sur toutes les plates-formes vxBLOB{100}=$04, vxBLOB{101}=$03, vxBLOB{102}=$02, vxBLOB{103}=$01
• Les autres octets du BLOB sont inchangés

(5) Après l'exécution de ce code :

FIXER TAILLE BLOB (vxBlob;100)
vlOffset:=50
⇒ **ENTIER LONG VERS BLOB** (0x01020304;vxBlob;Ordre octets Macintosh;vlOffset)

• La taille de vxBlob est 100 K
• Sur toutes les plates-formes vxBLOB{50}=$01, vxBLOB{51}=$02, vxBLOB{52}=$03, vxBLOB{53}=$04
• Les autres octets du BLOB restent inchangés
• La variable vlOffset a été incrémentée de 4 (et est alors égale à 54)

Référence

BLOB vers entier, BLOB vers entier long, BLOB vers reel, BLOB vers texte, ENTIER VERS BLOB, REEL VERS BLOB, TEXTE VERS BLOB.

REEL VERS BLOB

BLOB

version 6.0

REEL VERS BLOB (réel; blob; formatRéel{; offset | *})

Paramètre	Type		Description
réel	Numérique	→	Valeur de type Réel à écrire dans le BLOB
blob	BLOB	→	BLOB devant recevoir la valeur Réel
formatRéel	Numérique	→	0 Format réel natif 1 Format réel étendu 2 Format réel double Macintosh 3 Format réel double Windows
offset | *	Variable | *	→	Offset (en octets) dans le BLOB ou * pour ajouter la valeur à la fin du BLOB
		←	Nouvel offset après l'écriture si * omis

Description

La commande REEL VERS BLOB écrit la valeur de type Réel (ou Numérique) réel dans le BLOB blob.

Le paramètre formatRéel fixe le format interne et l'ordre des octets ("byte ordering") de la valeur de type Réel à écrire. Vous passez une des constantes fournies par 4e Dimension :

Constante	Type	Valeur
Format réel natif	Entier long	0
Format réel étendu	Entier long	1
Format réel double Macintosh	Entier long	2
Format réel double PC	Entier long	3

Note sur l'indépendance de plate-forme : Si vous échangez des BLOBs entre les plates-formes Macintosh et PC, il vous incombe de traiter les conversions d'octets ("byte swapping") lorsque vous utilisez cette fonction.

Si vous passez le paramètre optionnel *, la valeur réelle est ajoutée à la fin du BLOB et la taille de blob est modifiée en conséquence. Ainsi, à l'aide du paramètre optionnel *, vous pouvez stocker les unes derrière les autres autant de valeurs de type Entier, Entier long, Numérique ou Texte (référez-vous aux autres commandes sur les BLOBs) que vous voulez dans un BLOB, la seule limite étant celle de la mémoire disponible.

Si vous ne passez pas le paramètre optionnel * ni de variable dans le paramètre offset, la valeur réelle est stockée au début de blob en remplaçant son contenu précédent, et la taille du BLOB est modifiée en conséquence.

Si vous passez une variable dans le paramètre offset, le réel est écrit à partir de l'offset offset, exprimé en octets (à partir de zéro), du BLOB. Quel que soit l'endroit où vous placez la valeur, la taille du BLOB sera augmentée si nécessaire en fonction de l'emplacement que vous avez défini (plus jusqu'à 8 ou 10 octets le cas échéant). Les octets redéfinis (autres que ceux que vous venez d'écrire) sont initialisés à la valeur zéro.

Après l'exécution de la commande, la variable du paramètre offset est incrémentée du nombre d'octets ayant été écrits. Vous pouvez par conséquent réutiliser la même variable avec une autre commande d'écriture de BLOB afin de placer une autre valeur juste après celle que vous venez d'écrire.

Exemples

(1) Après l'exécution de ce code :

```
    C_REEL (vrValeur)
    vrValeur := ...
⇒   REEL VERS BLOB (vrValeur;vxBlob;Format réel natif)
```

• Sur PC et Power Macintosh, la taille de vxBlob est 8 octets
• Sur Macintosh 68K, la taille de vxBlob est 10 octets

(2) Après l'exécution de ce code :

```
    C_REEL (vrValeur)
    vrValeur := ...
⇒   REEL VERS BLOB (vrValeur;vxBlob;Format réel étendu)
```

• Sur toutes les plates-formes, la taille de vxBlob est 10 octets

(3) Après l'exécution de ce code :

```
    C_REEL (vrValeur)
    vrValeur := ...
⇒   REEL VERS BLOB (vrValeur;vxBlob;Format réel double Macintosh)
        ` ou Format réel double PC
```

• Sur toutes les plates-formes, la taille de vxBlob est 8 octets

(4) Après l'exécution de ce code :

```
FIXER TAILLE BLOB (vxBlob;100)
C_REEL (vrValeur)
vrValeur := ...
⇒ REEL VERS BLOB (vrValeur;vxBlob;Format réel double PC)
    ` ou Format réel double Macintosh
```

• Sur toutes les plates-formes, la taille de vxBlob est 8 octets

(5) Après l'exécution de ce code :

```
FIXER TAILLE BLOB (vxBlob;100)
⇒ REEL VERS BLOB (vrValeur;vxBlob;Format réel étendu;*)
```

• Sur toutes les plates-formes, la taille de vxBlob est 110 octets
• Sur toutes les plates-formes, la valeur numérique est stockée dans les octets #100 à #109
• Les autres octets du BLOB restent inchangés

(6) Après l'exécution de ce code :

```
FIXER TAILLE BLOB (vxBlob;100)
C_REEL (vrValeur)
vrValeur := ...
vlOffset:=50
⇒ REEL VERS BLOB (vrValeur;vxBlob;Format réel double PC;vlOffset)
    ` ou Format réel double Macintosh
```

• Sur toutes les plates-formes, la taille de vxBlob est 100 octets
• Sur toutes les plates-formes, la valeur numérique est stockée dans les octets #50 à #57
• Les autres octets du BLOB restent inchangés
• La variable vlOffset est incrémentée de 8 (et est alors égale à 58)

Référence

BLOB vers entier, BLOB vers entier long, BLOB vers reel, BLOB vers texte, ENTIER LONG VERS BLOB, ENTIER VERS BLOB, TEXTE VERS BLOB.

TEXTE VERS BLOB

BLOB

version 6.0

TEXTE VERS BLOB (texte; blob; formatTexte{; offset | *})

Paramètre	Type		Description
texte	Alpha	→	Valeur de type Texte à écrire dans le BLOB
blob	BLOB	→	BLOB devant recevoir la valeur de type Texte
formatTexte	Numérique	→	0 Chaîne en C 1 Chaîne pascal 2 Texte avec longueur 3 Texte sans longueur
offset | *	Variable | *	→	Offset (en octets) dans le BLOB ou * pour ajouter la valeur à la fin du BLOB
		←	Nouvel offset après l'écriture si * omis

Description

La commande TEXTE VERS BLOB écrit la valeur de type Texte texte dans le BLOB blob.

Le paramètre formatTexte définit le format interne de la valeur de type Texte à écrire. Vous passez une des constantes fournies par 4e Dimension :

Constante	Type	Valeur
Chaîne en C	Entier long	0
Chaîne pascal	Entier long	1
Texte avec longueur	Entier long	2
Texte sans longueur	Entier long	3

Le tableau suivant décrit chacun de ces formats :

Format texte	Description & Exemples
Chaîne en C	Le texte se termine par un caractère NULL (code ASCII $00). "" → $00 "Hello World!" → $48 65 6C 6C 6F 20 57 6F 72 6C 64 21 00
Chaîne pascal	Le texte est précédé d'un octet de longueur. "" → $00 "Hello World!" → $0C 48 65 6C 6C 6F 20 57 6F 72 6C 64 21

Texte avec longueur	Le texte est précédé de 2 octets de longueur. "" → $00 00 "Hello World!" → $00 0C 48 65 6C 6C 6F 20 57 6F 72 6C 64 21
Texte sans longueur	Le texte est composé seulement de ses caractères. "" → Pas de valeur "Hello World!" → $48 65 6C 6C 6F 20 57 6F 72 6C 64 21

Note : Cette commande accepte des expressions de type Texte (déclarées avec C_TEXTE) et de type Alpha (déclarées avec C_ALPHA). Notez bien que les variables de type Texte peuvent contenir jusqu'à 32 000 caractères et que les variables de type Alpha peuvent contenir un nombre de caractères fixe, défini lors de leur déclaration (255 caractères maximum).

Si vous passez le paramètre optionnel *, la valeur de type Texte est ajoutée à la fin du BLOB et la taille de blob est modifiée en conséquence. Ainsi, à l'aide du paramètre optionnel *, vous pouvez stocker les unes derrière les autres autant de valeurs de type Entier, Entier long, Numérique ou Texte (référez-vous aux autres commandes sur les BLOBs) que vous voulez dans un BLOB, la seule limite étant celle de la mémoire disponible.

Si vous ne passez pas le paramètre optionnel * ni de variable dans le paramètre offset, la valeur de type Texte est stockée au début de blob en remplaçant son contenu précédent, et la taille du BLOB est modifiée en conséquence.

Si vous passez une variable dans le paramètre offset, la valeur de type Texte est écrite à l'offset offset, exprimé en octets (à partir de zéro), du BLOB. Quel que soit l'endroit où vous placez la valeur, la taille du BLOB sera augmentée si nécessaire en fonction de l'emplacement que vous avez défini (plus jusqu'à la taille du texte le cas échéant). Les octets redéfinis (autres que ceux que vous venez d'écrire) sont initialisés à la valeur zéro.

Après l'exécution de la commande, la variable du paramètre offset est incrémentée du nombre d'octets ayant été écrits. Vous pouvez par conséquent réutiliser la même variable avec une autre commande d'écriture de BLOB afin de placer une autre valeur juste après celle que vous venez d'écrire.

Exemple

Après l'exécution de ce code :

```
     FIXER TAILLE BLOB (vxBlob;0)
     C_TEXTE (vtValeur)
     vtValeur := "Hello World!"  ` La longueur de vtValeur est de 12 octets
⇒    TEXTE VERS BLOB (vtValeur;Chaîne en C)  ` La taille du BLOB devient 13 octets
⇒    TEXTE VERS BLOB (vtValeur;Chaîne pascal)  ` La taille du BLOB devient 13 octets
⇒    TEXTE VERS BLOB (vtValeur;Texte avec longueur)
        ` La taille du BLOB devient 14 octets
⇒    TEXTE VERS BLOB (vtValeur;Texte sans longueur)  ` La taille du BLOB devient 12 octets
```

Référence

BLOB vers entier, BLOB vers entier long, BLOB vers reel, BLOB vers texte, ENTIER LONG VERS BLOB, ENTIER VERS BLOB, REEL VERS BLOB.

BLOB vers entier

BLOB

version 6.0

BLOB vers entier (blob; ordreOctet{; offset}) → Numérique

Paramètre	Type		Description
blob	BLOB	→	BLOB duquel obtenir la valeur entière
ordreOctet	Numérique	→	0 Ordre d'octets mode natif 1 Ordre d'octets Macintosh 2 Ordre d'octets PC
offset	Variable	→	Offset (en octets) dans le BLOB
		←	Nouvel offset après la lecture
Résultat	Numérique	←	Valeur entière (2 octets)

Description

BLOB vers entier retourne une valeur entière (2 octets) lue dans le BLOB blob.

Le paramètre ordreOctet fixe l'ordre des octets ("byte ordering") de la valeur entière à lire. Vous pouvez passer une des constantes fournies par 4e Dimension :

Constante	Type	Valeur
Ordre octets natif	Entier long	0
Ordre octets Macintosh	Entier long	1
Ordre octets PC	Entier long	2

Note sur l'indépendance de plate-forme : Si vous échangez des BLOBs entre les plates-formes Macintosh et PC, il vous incombe de traiter les conversions d'octets ("byte swapping") lorsque vous utilisez cette fonction.

Si vous ne passez pas de variable dans le paramètre optionnel offset, les deux premiers octets de BLOB sont lus.

Si vous passez une variable dans le paramètre optionnel offset, la valeur entière sur 2 octets est lue depuis l'offset exprimé en octets (à partir de zéro) du BLOB.

Note : Vous devez passer un offset compris entre 0 (zéro) et la taille du BLOB moins 2. Sinon, une erreur -111 est générée.

Après l'exécution de la commande, la variable offset est incrémentée du nombre d'octets qui a été lu. Vous pouvez donc réutiliser la même variable avec une autre commande de lecture de BLOBs pour lire une autre valeur placée juste après celle que vous venez de lire.

Exemple

L'exemple suivant lit 20 valeurs entières d'un BLOB à partir de l'offset 0x200 :

```
$vlOffset:=0x200
Boucle ($viBoucle;0;19)
⇒    $viValeur:=BLOB vers entier(vxUnBlob;Ordre octets PC;$vlOffset)
        ` Faire quelque chose avec $viValeur
Fin de boucle
```

Référence

BLOB vers entier long, BLOB vers reel, BLOB vers texte, ENTIER LONG VERS BLOB, ENTIER VERS BLOB, REEL VERS BLOB, TEXTE VERS BLOB.

BLOB vers entier long

BLOB

version 6.0

BLOB vers entier long (blob; ordreOctet{; offset}) → Numérique

Paramètre	Type		Description
blob	BLOB	→	BLOB duquel extraire la valeur de type Entier long
ordreOctet	Numérique	→	0 = Ordre d'octets natif 1 = Ordre d'octets Macintosh 2 = Ordre d'octets PC
offset	Variable	→	Offset (en octets) dans le BLOB
		←	Nouvel offset après lecture
Résultat	Numérique	←	Valeur de type Entier long (4 octets)

Description

La fonction BLOB vers entier long retourne une valeur de type Entier long (4 octets) lue dans le BLOB blob.

Le paramètre ordreOctet fixe l'ordre des octets ("byte ordering") de la valeur Entier long à lire. Vous passez une des constantes fournies par 4e Dimension :

Constante	Type	Valeur
Ordre octets natif	Entier long	0
Ordre octets Macintosh	Entier long	1
Ordre octets PC	Entier long	2

Note sur l'indépendance de plate-forme : Si vous échangez des BLOBs entre les plates-formes Macintosh et PC, il vous incombe de traiter les conversions d'octets ("byte swapping") lorsque vous utilisez cette fonction.

Si vous ne passez pas de variable dans le paramètre optionnel offset, les quatre premiers octets de BLOB sont lus.

Si vous passez une variable dans le paramètre optionnel offset, l'entier long sur 4 octets est lu depuis l'offset exprimé en octets (à partir de zéro) du BLOB.

Note : Vous devez passer un offset compris entre 0 (zéro) et la taille du BLOB moins 4. Sinon, une erreur -111 est générée.

Après l'exécution de la commande, la variable offset est incrémentée du nombre d'octets qui a été lu. Vous pouvez donc réutiliser la même variable avec une autre commande de lecture de BLOBs pour lire une autre valeur placée juste après celle que vous venez de lire.

Exemple

L'exemple suivant lit 20 valeurs de type Entier long dans un BLOB, à partir de l'offset 0x200 :

```
$vlOffset:=0x200
Boucle ($viBoucle;0;19)
⇒    $vlValeur:=BLOB vers entier long(vxUnBlob;Ordre octets PC;$vlOffset)
        ` Faire quelque chose avec $vlValeur
Fin de boucle
```

Référence

BLOB vers entier, BLOB vers reel, BLOB vers texte, ENTIER LONG VERS BLOB, ENTIER VERS BLOB, REEL VERS BLOB, TEXTE VERS BLOB.

BLOB vers reel

BLOB

version 6.0

BLOB vers reel (blob; formatRéel{; offset}) → Numérique

Paramètre	Type		Description
blob	BLOB	→	BLOB duquel extraire la valeur de type Réel
formatRéel	Numérique	→	0 Format réel natif 1 Format réel étendu 2 Format réel double Macintosh 3 Format réel double Windows
offset	Variable	→	Offset (en octets) dans le BLOB
		←	Nouvel offset après lecture
Résultat	Numérique	←	Valeur de type Réel

Description

La fonction BLOB vers reel retourne une valeur de type Réel (ou Numérique) lue dans le BLOB blob.

Le paramètre formatRéel fixe le format interne et l'ordre des octets ("byte ordering") de la valeur de type Réel à lire. Vous passez une des constantes fournies par 4e Dimension :

Constante	Type	Valeur
Format réel natif	Entier long	0
Format réel étendu	Entier long	1
Format réel double Macintosh	Entier long	2
Format réel double PC	Entier long	3

Note sur l'indépendance de plate-forme : Si vous échangez des BLOBs entre les plates-formes Macintosh et PC, il vous incombe de traiter les conversions d'octets ("byte swapping") lorsque vous utilisez cette fonction.

Si vous ne passez pas de variable dans le paramètre optionnel offset, les 8 ou 10 premiers octets de BLOB sont lus.

Si vous passez une variable dans le paramètre optionnel offset, la valeur réelle est lue depuis l'offset exprimé en octets (à partir de zéro) du BLOB.

Note : Vous devez passer un offset compris entre 0 (zéro) et la taille du BLOB moins 8 ou 10. Sinon, une erreur -111 est générée.

Après l'exécution de la commande, la variable offset est incrémentée du nombre d'octets qui a été lu. Vous pouvez donc réutiliser la même variable avec une autre commande de lecture de BLOBs pour lire une autre valeur placée juste après celle que vous venez de lire.

Exemple

L'exemple suivant lit 20 valeurs réelles dans un BLOB à partir de l'offset 0x200 :

```
$vlOffset:=0x200
Boucle ($viBoucle;0;19)
⇒    $vrValeur:=BLOB vers reel(vxUnBlob;Format réel double PC;$vlOffset)
     ` Faire quelque chose avec $vrValeur
Fin de boucle
```

Référence

BLOB vers entier, BLOB vers entier long, BLOB vers texte, ENTIER LONG VERS BLOB, ENTIER VERS BLOB, REEL VERS BLOB, TEXTE VERS BLOB.

BLOB vers texte

BLOB

version 6.0

BLOB vers texte (blob; formatTexte{; offset{; longueurTexte}}) → Chaîne

Paramètre	Type		Description
blob	BLOB	→	BLOB duquel extraire la valeur de type Texte
formatTexte	Numérique	→	0 Chaîne en C 1 Chaîne pascal 2 Texte avec longueur 3 Texte sans longueur
offset	Variable	→	Offset (en octets) dans le BLOB
		←	Nouvel offset après la lecture
longueurTexte	Numérique	→	Nombre de caractères à lire
Résultat	Chaîne	←	Valeur de type Texte

Description

La fonction BLOB vers texte retourne une valeur de type Texte lue dans le BLOB blob.

Le paramètre formatTexte définit le format interne de la valeur de type Texte à écrire. Vous passez une des constantes fournies par 4e Dimension :

Constante	Type	Valeur
Chaîne en C	Entier long	0
Chaîne pascal	Entier long	1
Texte avec longueur	Entier long	2
Texte sans longueur	Entier long	3

Le tableau suivant décrit chacun de ces formats :

Format texte	Description & Exemples
Chaîne en C	Le texte se termine par un caractère NULL (code ASCII $00). "" → $00 "Hello World!" → $48 65 6C 6C 6F 20 57 6F 72 6C 64 21 00
Chaîne pascal	Le texte est précédé d'un octet de longueur. "" → $00 "Hello World!" → $0C 48 65 6C 6C 6F 20 57 6F 72 6C 64 21

Texte avec longueur	Le texte est précédé de 2 octets de longueur. "" → $00 00 "Hello World!" → $00 0C 48 65 6C 6C 6F 20 57 6F 72 6C 64 21
Texte sans longueur	Le texte est composé seulement de ses caractères. "" → Pas de valeurs "Hello World!" → $48 65 6C 6C 6F 20 57 6F 72 6C 64 21

ATTENTION : Le nombre de caractères à lire est déterminé par le paramètre formatTexte, SAUF dans le cas du dernier format, Texte sans longueur, pour lequel vous devez spécifier le nombre de caractères à lire dans le paramètre longueurTexte. Pour les autres formats, longueurTexte est ignoré et vous pouvez l'omettre.

Notez bien que les variables de type Texte peuvent contenir jusqu'à 32 000 caractères et que les variables de type Alpha peuvent contenir un nombre de caractères fixe, défini lors de leur déclaration (255 caractères maximum). Si vous tentez de lire plus de valeurs que la variable ne peut en contenir, 4D tronque le résultat de la commande.

Si vous ne passez pas de variable dans le paramètre optionnel offset, les premiers octets de BLOB sont lus, en fonction de la valeur passée dans formatTexte. Notez que vous devez passer une variable dans le paramètre offset lorsque vous lisez une valeur de type Texte sans longueur.

Si vous passez une variable dans le paramètre optionnel offset, la valeur de type Texte est lue depuis l'offset exprimé en octets (à partir de zéro) du BLOB.

Note : Vous devez passer un offset compris entre 0 (zéro) et la taille du BLOB moins la taille du texte à extraire. Sinon, le résultat de la fonction ne sera pas exploitable.

Exemple

L'exemple suivant lit une ressource type MacOS imaginaire dont le format interne est identique à celui des ressources 'STR#' :

```
LIRE RESSOURCE ("ABCD";viRéfRes;vxDonnéesRes;viMonFichierRes)
vlTaille:=Taille BLOB(vxDonnéesRes)
Si (vlTaille>0)
      ` La ressource commence par un entier sur 2 octets qui spécifie le nombre de
      ` chaînes
   vlOffset:=0
   viNbSaisies:=BLOB vers entier(vxDonnéesRes;Ordre octets Macintosh;vlOffset)
      ` Ensuite la ressource contient des chaînes Pascal concaténées sans octet de
      ` réalignement
   Boucle (viSaisie;1;viNbSaisies)
      Si (vlOffset<vlTaille)
⇒        vaSaisie:=BLOB vers texte(vxDonnéesRes;Chaîne pascal;vlOffset)
            ` Faire quelque chose avec vaSaisie
      Sinon
            ` Les données de la ressource sont invalides, sortir de la boucle
         viSaisie:=viNbSaisies+1
      Fin de si
   Fin de boucle
Fin de si
```

Référence

BLOB vers entier, BLOB vers entier long, BLOB vers reel, ENTIER LONG VERS BLOB, ENTIER VERS BLOB, REEL VERS BLOB, TEXTE VERS BLOB.

INSERER DANS BLOB

BLOB

version 6.0

INSERER DANS BLOB (blob; offset; nombre{; défaut})

Paramètre	Type		Description
blob	BLOB	→	BLOB dans lequel insérer les octets
offset	Variable	→	Position de début d'insertion des octets
nombre	Numérique	→	Nombre d'octets à insérer
défaut	Numérique	→	Valeur d'octet par défaut (0x00..0xFF) 0x00 si ce paramètre est omis

Description

INSERER DANS BLOB insère le nombre d'octets spécifié par nombre dans le BLOB blob à la position spécifiée par offset. La taille du BLOB est augmentée de nombre d'octets.

Si vous ne passez pas le paramètre optionnel défaut, la valeur des octets insérés dans le BLOB est fixée à 0x00. Sinon, les octets prennent la valeur passée dans défaut (modulo 256 — 0..255).

Vous passez dans la variable du paramètre offset la position (relative à l'origine du BLOB) de l'insertion.

Référence

SUPPRIMER DANS BLOB.

SUPPRIMER DANS BLOB

BLOB

version 6.0

SUPPRIMER DANS BLOB (blob; offset; nombre)

Paramètre	Type		Description
blob	BLOB	→	BLOB duquel supprimer des octets
offset	Numérique	→	Offset à partir duquel supprimer les octets
nombre	Numérique	→	Nombre d'octets à supprimer

Description

SUPPRIMER DANS BLOB supprime le nombre d'octets spécifié par nombre du BLOB blob à partir de la position définie par offset (exprimée de manière relative à l'origine du BLOB). La taille du BLOB est réduite de nombre d'octets.

Référence

INSERER DANS BLOB.

COPIER BLOB

BLOB

version 6.0

COPIER BLOB (srcBLOB; dstBLOB; srcOffset; dstOffset; nombre)

Paramètre	Type		Description
srcBLOB	BLOB	→	BLOB source
dstBLOB	BLOB	→	BLOB de destination
srcOffset	Variable	→	Position dans la source pour la copie
dstOffset	Variable	→	Position dans la destination pour la copie
nombre	Numérique	→	Nombre d'octets à copier

Description

COPIER BLOB copie le nombre d'octets spécifié par nombre du BLOB srcBLOB vers le BLOB dstBLOB.

La copie commence à la position (exprimée par rapport à l'origine du BLOB source) définie par srcOffset et est placée à partir de la position (exprimée par rapport à l'origine du BLOB de destination) définie par dstOffset.

Notez que le BLOB de destination peut être redimensionné si nécessaire.

Référence

INSERER DANS BLOB, SUPPRIMER DANS BLOB.

CRYPTER BLOB

BLOB

version 6.7

CRYPTER BLOB (aCrypter; cléPrivEmetteur{; cléPubRécepteur})

Paramètre	Type		Description
aCrypter	BLOB	→	Données à crypter
		←	Données cryptées
cléPrivEmetteur	BLOB	→	Clé privée de l'émetteur
cléPubRécepteur	BLOB	→	Clé publique du récepteur

Description

La commande CRYPTER BLOB permet de crypter le contenu du BLOB aCrypter à l'aide de la clé privée de l'émetteur cléPrivEmetteur ainsi que, optionnellement, de la clé publique du récepteur cléPubRécepteur. Pour obtenir une paire de clés de cryptage (clé publique et clé privée), utilisez la routine GENERER CLES CRYPTAGE, placée dans le thème "Protocole sécurisé".

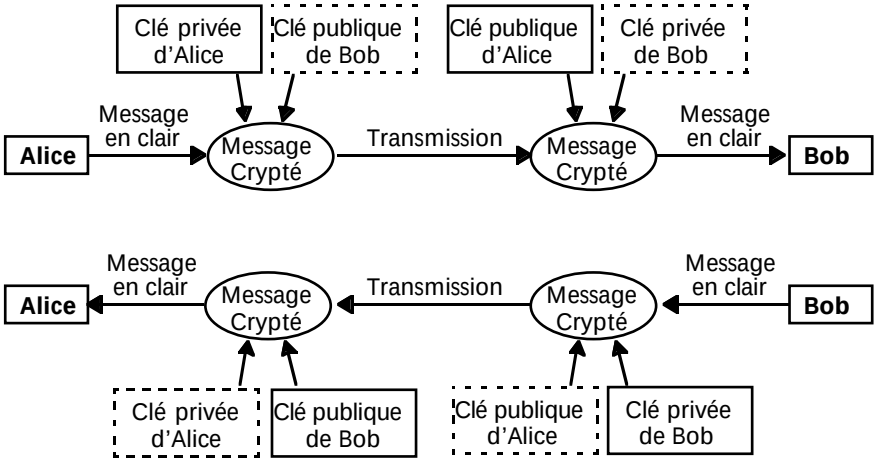
Note : La commande CRYPTER BLOB exploite l'algorithme et les fonctions de cryptage du protocole SSL. Par conséquent, pour pouvoir utiliser cette commande, vous devez veiller à ce que les composants nécessaires au fonctionnement du protocole SSL soient installés sur la machine — même si vous ne souhaitez pas utiliser SSL dans le cadre de connexions à un serveur Web 4D. Pour plus d'informations, reportez-vous à la section Services Web, Utiliser le protocole SSL.

• L'utilisation d'une seule clé pour le cryptage (clé privée de l'émetteur) garantit l'impossibilité pour toute personne ne disposant pas de la clé publique de lire les données. Elle garantit également que c'est bien l'émetteur qui a crypté les données.
• L'utilisation d'une paire de clés pour le cryptage (clé privée de l'émetteur + clé publique du récepteur) garantit en outre qu'un seul récepteur pourra lire les données.

Le format interne des BLOBs contenant des clés est le PEM (*Privacy Enhanced Mail*). Ce format, multi-plate-forme, permet l'échange ou la manipulation des clés par simple copier-coller dans un Email ou un fichier texte.

Après l'exécution de la commande, le BLOB aCrypter contient les données cryptées. Ces données ne pourront être décryptées qu'avec la commande DECRYPTER BLOB, à laquelle la clé publique de l'émetteur sera passée en paramètre. En outre, si la clé publique (optionnelle) du récepteur avait été utilisée pour le cryptage, la clé privée du récepteur sera également nécessaire pour le décryptage.

Principe du cryptage à clés publiques/privées pour l'échange de messages entre deux individus, "Alice" et "Bob" :



Note : L'algorithme de cryptage comporte une fonction de vérification d'intégrité (*checksum*), afin d'empêcher toute modification malveillante ou accidentelle du contenu du BLOB. Par conséquent, un BLOB crypté ne doit pas être modifié, sous peine de ne pas pouvoir être décrypté.

Optimisation des commandes de cryptage

Le cryptage des données ralentit l'exécution de l'application, en particulier si une paire de clés est utilisée. Deux types d'optimisations sont toutefois possibles :

• Suivant la quantité de mémoire disponible, la commande s'exécute en mode "synchrone" ou "asynchrone".
Le mode asynchrone est plus rapide, car il ne bloque pas les autres process. Ce mode est automatiquement utilisé si la mémoire disponible est au moins égale à 2 fois la taille de la source à crypter.
Dans le cas contraire, pour des raisons de sécurité, le mode synchrone est utilisé. Ce mode est plus lent car les autres process sont bloqués.

• Dans le cas de BLOBs volumineux, l'astuce consiste à crypter uniquement une partie déterminée et sensible du BLOB, afin de réduire la taille des données à traiter et donc le temps d'exécution.

Exemples

• Utilisation d'une seule clé

Une société veut garantir la confidentialité d'informations stockées dans une base 4D. Elle doit régulièrement envoyer ces données à ses filiales, par exemple sous la forme de fichiers via Internet.

1. La société commence par générer une paire de clés à l'aide de la commande GENERER CLES CRYPTAGE.

```
   `Méthode GENERE_CLES_TXT
C_BLOB($BcléPublique; $BcléPrivée)
GENERER CLES CRYPTAGE($BcléPrivée;$BcléPublique)
BLOB VERS DOCUMENT("cléPublique.txt"; $BcléPublique)
BLOB VERS DOCUMENT("cléPrivée.txt"; $BcléPrivée)
```

2. La société conserve la clé privée, et remet à chaque filiale une copie du document contenant la clé publique. Il faut, bien entendu, que cette transmission s'effectue d'une façon sûre, par exemple par la copie sur une disquette donnée physiquement aux filiales.

3. Par la suite, la société copie les informations confidentielles (stockées par exemple dans un champ texte) dans des BLOBs et les crypte avec sa clé privée :

```
   `Méthode CRYPTER_INFOS
C_BLOB($vbCrypté;$vbcléPrivée)
C_TEXTE($vtCrypter)

$vtCrypter:=[Confidentiel]Info
VARIABLE VERS BLOB ($vtCrypter;$vbCrypté)
DOCUMENT VERS BLOB("cléPrivée.txt"; $vbcléPrivée)
Si (OK=1)
⇒  CRYPTER BLOB ($vbCrypté; $vbcléPrivée)
   BLOB VERS DOCUMENT ("MiseAJour.txt";$vbCrypté)
Fin de si
```

4. Le fichier de mise à jour peut alors être envoyé aux filiales (même en passant par un canal non sécurisé comme Internet). Si un tiers intercepte le fichier crypté, il sera dans l'incapacité de le décrypter sans la clé publique.

5. Chaque filiale peut, quant à elle, décrypter le document à l'aide de la clé publique :

```
   `Méthode DECRYPTER_INFOS
C_BLOB($vbCrypté;$vbcléPublique)
C_TEXTE($vtDécrypté)
C_HEURE ($vhRefDoc)

ALERTE ("Veuillez sélectionner le document crypté.")
$vhRefDoc:=Ouvrir document("") `Sélection du fichier MiseAJour.txt
Si (OK=1)
   FERMER DOCUMENT($vhRefDoc)
   DOCUMENT VERS BLOB(Document;$vbCrypté)
```

```
        DOCUMENT VERS BLOB("cléPublique.txt"; $vbcléPublique)
        Si (OK=1)
⇒           DECRYPTER BLOB ($vbCrypté; $vbcléPublique)
            BLOB VERS VARIABLE($vbCrypté; $vtDécrypté)
            CREER ENREGISTREMENT ([Confidentiel])
            [Confidentiel]Info:=$vtDécrypté
            STOCKER ENREGISTREMENT([Confidentiel])
        Fin de si
    Fin de si
```

• Utilisation de deux clés

Une société souhaite utiliser un système d'échange de données via Internet dans lequel chaque filiale reçoit des informations confidentielles mais envoie également ses propres informations à la maison-mère. Ce système a donc les impératifs suivants :

- Seul le destinataire doit pouvoir lire un message,
- On doit avoir la garantie que le message provient bien de l'expéditeur.

1. La maison-mère ainsi que chaque filiale génèrent leurs propres paires de clés (à l'aide de la méthode *GENERE_CLES_TXT*).

```
    `Méthode GENERE_CLES_TXT
C_BLOB($BcléPublique; $BcléPrivée)
GENERER CLES CRYPTAGE($BcléPrivée;$BcléPublique)
BLOB VERS DOCUMENT("cléPublique.txt"; $BcléPublique)
BLOB VERS DOCUMENT("cléPrivée.txt"; $BcléPrivée)
```

2. Chacune garde sa clé privée. Chaque filiale envoie sa clé publique à la maison-mère, qui elle-même envoie sa clé publique à chaque filiale. Cette transmission ne doit pas nécessairement être effectuée par un canal protégé, car la seule détention de la clé publique dans ce cas sera insuffisante pour décrypter une information.

3. Pour crypter une information à envoyer, une filiale ou la maison-mère exécute la méthode *CRYPTER_INFOS_2* qui utilise la clé privée de l'émetteur et la clé publique du destinataire pour crypter les données :

```
    `Méthode CRYPTER_INFOS_2
C_BLOB($vbCrypté;$vbcléPrivée;$vbcléPublique)
C_TEXTE($vtCrypter)
C_HEURE ($vhRefDoc)

$vtCrypter:=[Confidentiel]Info
VARIABLE VERS BLOB ($vtCrypter;$vbCrypté)
   ` On charge sa propre clé privée...
DOCUMENT VERS BLOB("cléPrivée.txt"; $vbcléPrivée)
Si (OK=1)
      ` ...et la clé publique du récepteur
   ALERTE ("Veuillez sélectionner la clé publique du destinataire.")
   $vhRefDoc:=Ouvrir document("") `Sélection de la clé publique à charger
```

```
      Si (OK=1)
         FERMER DOCUMENT($vhRefDoc)
         DOCUMENT VERS BLOB(Document;$vbcléPublique)
            `Cryptage du BLOB avec les deux clés en paramètres
⇒        CRYPTER BLOB ($vbCrypté; $vbcléPrivée; $vbcléPublique)
         BLOB VERS DOCUMENT ("MiseAJour.txt";$vbCrypté)
      Fin de si
   Fin de si
```

4. Le fichier crypté peut alors être envoyé au destinataire via Internet. Si un tiers l'intercepte, il sera dans l'incapacité de le décrypter, même en connaissant les clés publiques, car il lui manquera la clé privée du destinataire.

5. Chaque destinataire peut, quant à lui, décrypter le document reçu, en utilisant sa clé privée et la clé publique de l'émetteur :

```
   `Méthode DECRYPTER_INFOS_2
C_BLOB($vbCrypté;$vbcléPublique;$vbcléPrivée)
C_TEXTE($vtDécrypté)
C_HEURE ($vhRefDoc)

ALERTE ("Veuillez sélectionner le document crypté.")
$vhRefDoc:=Ouvrir document("") `Sélection du fichier MiseAJour.txt
Si (OK=1)
   FERMER DOCUMENT($vhRefDoc)
   DOCUMENT VERS BLOB(Document;$vbCrypté)
      `On charge sa propre clé privée
   DOCUMENT VERS BLOB("cléPrivée.txt"; $vbcléPrivée)
   Si (OK=1)
      ` ...et la clé publique de l'émetteur
      ALERTE ("Veuillez sélectionner la clé publique de l'envoyeur.")
      $vhRefDoc:=Ouvrir document("") `Sélection de la clé publique
      Si (OK=1)
         FERMER DOCUMENT($vhRefDoc)
         DOCUMENT VERS BLOB(Document;$vbcléPublique)
            `Décryptage du BLOB avec les deux clés en paramètres
⇒        DECRYPTER BLOB ($vbCrypté; $vbcléPublique;$vbcléPrivée)
         BLOB VERS VARIABLE($vbCrypté; $vtDécrypté)
         CREER ENREGISTREMENT ([Confidentiel])
         [Confidentiel]Info:=$vtDécrypté
         STOCKER ENREGISTREMENT([Confidentiel])
      Fin de si
   Fin de si
Fin de si
```

Référence

DECRYPTER BLOB, GENERER CLES CRYPTAGE.

DECRYPTER BLOB

BLOB

version 6.7

DECRYPTER BLOB (aDécrypter; cléPubEmetteur{; cléPrivRécepteur})

Paramètre	Type		Description
aDécrypter	BLOB	→	Données à décrypter
		←	Données décryptées
cléPubEmetteur	BLOB	→	Clé publique de l'émetteur
cléPrivRécepteur	BLOB	→	Clé privée du récepteur

Description

La commande DECRYPTER BLOB permet de décrypter le contenu du BLOB aDécrypter à l'aide de la clé publique de l'émetteur cléPubEmetteur ainsi que, optionnellement, de la clé privée du récepteur cléPrivRécepteur.

Vous passez dans le paramètre cléPubEmetteur le BLOB contenant la clé publique de l'émetteur. Cette clé a été générée par l'émetteur (à l'aide de la commande GENERER CLES CRYPTAGE), qu'il doit ensuite transmettre au récepteur.

Le paramètre optionnel cléPrivRécepteur doit recevoir la clé privée du récepteur. Dans ce cas, le récepteur doit également avoir généré une paire de clés de cryptage à l'aide de GENERER CLES CRYPTAGE et transmis sa clé publique à l'émetteur. Le système de cryptage à deux clés permet de garantir que seul l'émetteur peut avoir crypté le message et seul le récepteur peut le décrypter.
Pour plus d'informations sur le système de cryptage à deux clés, reportez-vous à la description de la commande CRYPTER BLOB.

La commande DECRYPTER BLOB comporte une fonction de vérification d'intégrité (*checksum*), afin d'empêcher toute modification malveillante ou accidentelle du contenu du BLOB. Si le BLOB crypté est endommagé ou modifié, la commande ne fera rien et retournera une erreur.

Exemple

Reportez-vous aux exemples de la commande CRYPTER BLOB.

Référence

CRYPTER BLOB, GENERER CLES CRYPTAGE.

4

Booléens

Les fonctions booléennes de 4D sont utilisées pour les opérations de type booléennes, c'est-à-dire ne traitant que deux valeurs, VRAI ou FAUX. Ces fonctions sont les suivantes :

- Vrai
- Faux
- Non

L'exemple suivant retourne Vrai dans la variable monBooléen si l'utilisateur a cliqué sur le bouton monBouton et Faux s'il n'a pas cliqué dessus. Lorsqu'un bouton reçoit un clic, la variable du bouton prend la valeur 1. Dans cet exemple, la valeur de la variable booléenne est basée sur la valeur d'un bouton.

```
Si (monBouton=1)   ` Si le bouton a reçu un clic
   monBooléen:=Vrai   ` monBooléen prend la valeur Vrai
Sinon   ` Si le bouton n'a pas reçu de clic,
   monBooléen:=Faux   ` monBooléen prend la valeur Faux
Fin de si
```

L'exemple ci-dessus peut être simplifié et écrit en une seule ligne, plus efficace :

```
monBooléen:=(monBouton=1)
```

Référence

Faux, Non, Opérateurs logiques, Vrai.

Vrai

version 5

Vrai → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Vrai

Description

Vrai retourne la valeur booléenne Vrai.

Exemple

L'exemple suivant met la variable vbOptions à Vrai :

```
⇒ vbOptions:=Vrai
```

Référence

Faux, Non.

Faux

Faux → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Faux

Description

Faux retourne la valeur booléenne Faux.

Exemple

L'exemple suivant met la variable vbOptions à Faux :

```
⇒ vbOptions:=Faux
```

Référence

Non, Vrai.

Non (booléen) → Booléen

Paramètre	Type		Description
booléen	Booléen	→	Valeur booléenne à inverser
Résultat	Booléen	←	Inverse de booléen

Description

La fonction Non retourne la valeur inverse de booléen, changeant un Vrai en Faux ou un Faux en Vrai.

Exemples

Dans l'exemple suivant, la valeur Vrai est assignée à une variable. Cette valeur est alors modifiée en Faux puis de nouveau en Vrai :

```
Résultat:= Vrai ` Résultat prend la valeur VRAI
Résultat:= Non (Résultat) ` Résultat prend la valeur FAUX
Résultat:= Non (Résultat) ` Résultat prend la valeur VRAI
```

Référence

Faux, Vrai.

5

Chaînes de caractères

Chaine (expression{; format}) → Alpha

Paramètre	Type		Description
expression	type	→	Expression à convertir en chaîne (peut être de Numérique, Entier, Entier long, Date ou Heure)
format	Alpha | Num	→	Format d'affichage
Résultat	Alpha	←	expression convertie en chaîne alphanumérique

Description

La commande Chaine retourne sous forme de chaîne alphanumérique l'expression de type numérique, Date ou Heure, que vous avez passée dans le paramètre expression.

Si vous ne passez pas le paramètre optionnel format, la chaîne est retournée dans le format par défaut du type de données correspondant. Si vous passez le paramètre format, vous pouvez définir suivant vos besoins le formatage de la chaîne retournée.

Expressions numériques
Si expression est du type numérique (Réel, Entier, Entier long), vous pouvez passer le paramètre optionnel de formatage de la chaîne. Voici quelques exemples :

Exemple	Résultat
Chaine(2^15) `Utiliser format défaut	32768 (Format par défaut)
Chaine(2^15;"### ##0 habitants")	32 768 habitants
Chaine(1/3;"##0,00000")	0,33333
Chaine(1/3) `Utiliser format défaut	0,3333333333333333 (Format défaut)
Chaine(Arctan(1)*4)	3,1415926535897931 (Format défaut)
Chaine(Arctan(1)*4;"##0,00")	3,14
Chaine(-1;"&x")	0xFFFFFFFF
Chaine(-1;"&$")	$FFFFFFFF
Chaine(0 ?+ 7;"&x")	0x80
Chaine(0 ?+ 7;"&$")	$80
Chaine(0 ?+ 14;"&x")	0x4000
Chaine(0 ?+ 14;"&$")	$4000
Chaine(Num(1=1);"Vrai;;Faux")	Vrai
Chaine(Num(1=2);"Vrai;;Faux")	Faux

Le format est défini de la même manière que pour un champ numérique dans un formulaire. Pour plus d'informations sur le formatage des numériques, reportez-vous au manuel *Mode Structure* de 4e Dimension. Vous pouvez également passer le nom d'un style personnalisé dans format. Dans ce cas, le nom du style doit être précédé du caractère "|".

Expressions de type Date
Si expression est de type Date, la chaîne est retournée dans le format par défaut du pays (par exemple, JJ-MM-AA pour la version française). Vous pouvez passer une valeur dans le paramètre format, en fonction du tableau ci-dessous :

Numéro	Format	Exemple
1	Court	06-12-96
2	Abrégé	ven 6 déc 1996
3	Long	vendredi 6 décembre 1996
4	Spécial	06-12-96 (mais 06-12-1896 ou 06-12-2096)
5	Jour Mois Année	6 décembre 1996
6	Abrégé Jour Mois Année	6 déc 1996
7	Spécial forcé	06-12-1996

4D fournit les constantes prédéfinies suivantes :

Constante	Type	Valeur
Format court	Entier long	1
Format abrégé	Entier long	2
Format long	Entier long	3
Format spécial	Entier long	4
Jour Mois Année	Entier long	5
Abrégé Jour Mois Année	Entier long	6
Spécial forcé	Entier long	7

Voici quelques exemples (en supposant que nous soyons mardi 4 mars 1997) :

```
$vsRésultat:=Chaine(Date du jour)  ` $vsRésultat prend la valeur "04-03-97"
$vsRésultat:=Chaine(Date du jour;Jour Mois Année)
   ` $vsRésultat prend la valeur "4 Mars 1997"
```

Expressions de type Heure
Si expression est de type Heure, la chaîne est retournée dans le format par défaut hh:mm:ss. Vous pouvez passer une valeur dans le paramètre format, en fonction du tableau ci-dessous :

Numéro	Format	Exemple
1	hh:mm:ss	01:02:03
2	hh:mm	01:02
3	heure minute seconde	1 heure 2 minutes 3 secondes
4	heure minute	1 heure 2 minutes
5	h:mm matin/après-midi	1:02 du matin

4D fournit les constantes suivantes :

Constante	Type	Valeur
h mn s	Entier long	1
h mn	Entier long	2
Heure Minute Seconde	Entier long	3
Heure Minute	Entier long	4
h mn Matin Après Midi	Entier long	5

Voici quelques exemples (en supposant qu'il soit 17h30 et 45 secondes) :

⇒ $vsRésultat:=**Chaine**(**Heure courante**) ` $vsRésultat prend la valeur "17:30:45"
⇒ $vsRésultat:=**Chaine**(**Heure courante**;Heure Minute Seconde)
` $vsRésultat prend la valeur "17 heures 30 minutes 45 secondes"

Référence
Chaine heure, Date, Num.

Num (expression) → Numérique

Paramètre	Type		Description
expression	Alpha | Booléen	→	Chaîne à convertir en numérique ou Booléen à convertir en 0 ou 1
Résultat	Numérique	←	Valeur numérique d'une chaîne ou d'un booléen

Description

La fonction Num retourne sous forme de numérique l'expression de type Alpha ou Booléen que vous avez passée dans le paramètre expression.

Expressions de type Alpha
Si expression ne contient que des caractères alphabétiques, Num retourne zéro. Si expression contient des caractères alphabétiques et des caractères numériques, Num ignore les caractères alphabétiques. Ainsi, Num transformera la chaîne "a1b2c3" en nombre 123.

Note : Seuls les 32 premiers caractères de expression sont pris en compte.

Il existe trois caractères réservés que Num traite de manière particulière. Il s'agit du sépateur décimal (c'est-à-dire la virgule (,) dans une version française), du tiret (-) et du e (ou E). Ils seront interprétés en tant que caractères de formatage des nombres :

• Le séparateur décimal est interprété en tant que tel et doit être inclus dans une chaîne de caractères numériques.
• Le tiret définit un nombre ou un exposant négatif (signe moins). Le tiret doit être placé devant tout caractère numérique négatif ou derrière le e pour un exposant. Si le tiret est inclus dans une chaîne numérique, la chaîne est considérée comme invalide et la fonction Num retourne zéro. Par exemple, Num(123-456) retourne 0, mais Num(-9) retourne -9.
• Le e ou E désigne tout caractère numérique se trouvant à sa droite comme étant la puissance d'un exposant. Le e doit être inclus dans une chaîne numérique. Ainsi, Num ("123e–2") retourne 1,23.

Expressions de type Booléen
Si vous passez une expression booléenne dans le paramètre expression, Num retourne 1 si expression est VRAI, sinon Num retourne 0.

Exemples

(1) L'exemple suivant illustre la manière dont Num fonctionne lorsqu'un argument numérique lui est passé. A chaque ligne, un numérique est assigné à la variable vRésultat. Les commentaires décrivent les résultats :

```
⇒    vRésultat := Num ("ABCD") ` vRésultat vaut 0
⇒    vRésultat := Num ("A1B2C3") ` vRésultat vaut 123
⇒    vRésultat := Num ("123") ` vRésultat vaut 123
⇒    vRésultat := Num ("123,4") ` vRésultat vaut 123,4
⇒    vRésultat := Num ("–123") ` vRésultat vaut –123
⇒    vRésultat := Num ("–123e2") ` vRésultat vaut –12300
```

(2) Dans l'exemple suivant, [Client]Dette est comparé à la valeur 5000 (francs). La fonction Num appliquée à cette comparaison retourne 0 ou 1. La multiplication d'une chaîne par 0 ou 1 retourne soit la chaîne, soit une chaîne vide. En définitive, le champ [Client]Risque reçoit la valeur “Acceptable” ou “Inacceptable” :

```
      ` Si le client a des dettes inférieures à 5000, le risque est acceptable.
      ` Si le client a des dettes supérieures à 5000, le risque est inacceptable.
⇒    [Client]Risque:=("Acceptable"* Num([Client]Dettes < 5000))+("Inacceptable" *
                                                    Num([Client]Dettes >= 5000))
```

Référence

Chaine, Opérateurs logiques, Opérateurs sur les chaînes.

Position (àChercher; chaîne) → Numérique

Paramètre	Type		Description
àChercher	Alpha	→	Chaîne à rechercher
chaîne	Alpha	→	Chaîne dans laquelle effectuer la recherche
Résultat	Numérique	←	Position de la première occurrence de àChercher

Description

Position retourne la position de la première occurrence de àChercher dans chaîne.

Si chaîne ne contient pas àChercher, la fonction retourne zéro (0).

Si Position trouve une occurrence de àChercher, la fonction retourne la position du premier caractère de cette occurrence dans chaîne.

Si vous demandez la position d'une chaîne vide à l'intérieur d'une chaîne vide, Position retourne zéro (0).

Attention : Vous ne pouvez pas utiliser le caractère joker (@) avec Position. Si, par exemple, vous passez "abc@" dans àChercher, la fonction recherchera effectivement la chaîne "abc@" et non pas "abc suivi de toute valeur".

Exemple

L'exemple suivant illustre l'utilisation de Position. Les résultats sont assignés à la variable vRésultat. Les commentaires fournissent la valeur de vRésultat :

```
⇒ vRésultat := Position ("ll"; "Billard") ` vRésultat prend la valeur 3
     ` Position de la première occurrence de vText1 dans vText2
⇒ vRésultat := Position (vText1; vText2)
```

Référence

Opérateurs de comparaison, Sous chaine.

Sous chaine (source; àPartirDe{; nbCars}) → Alpha

Paramètre	Type		Description
source	Alpha	→	Chaîne de laquelle extraire une sous-chaîne
àPartirDe	Numérique	→	Position du premier caractère
nbCars	Numérique	→	Nombre de caractères à extraire
Résultat	Alpha	←	Sous-chaîne de source

Description

La fonction Sous chaine retourne la partie de source délimitée par les paramètres àPartirDe et nbCars.

Le paramètre àPartirDe indique le premier caractère de la chaîne à retourner, et nbCars définit le nombre de caractères à retourner.

Si nbCars n'est pas défini ou si le total de àPartirDe plus nbCars est supérieur au nombre de caractères de la chaîne source, Sous chaine retourne tous les caractères de la chaîne à partir du caractère spécifié par àPartirDe. Si àPartirDe est supérieur au nombre de caractères de la chaîne, Sous chaine retourne une chaîne vide ("").

Exemples

(1) L'exemple suivant illustre l'utilisation de Sous chaine. Les résultats sont assignés à la variable vRésultat. Les commentaires fournissent la valeur de vRésultat :

```
vRésultat := Sous chaine ("08/04/62"; 4; 2) ` vRésultat prend la valeur "04"
vRésultat := Sous chaine ("Important"; 1; 6) ` vRésultat prend la valeur "Import"
vRésultat := Sous chaine (var; 2) ` vRésultat retourne tous les caractères sauf le premier
```

(2) La méthode projet suivante ajoute au tableau de type texte ou alpha, dont le pointeur est passé en second paramètre, les paragraphes tirés du texte passé en premier paramètre :

```
    ` EXTRAIRE PARAGRAPHES
    ` EXTRAIRE PARAGRAPHES ( Texte ; Pointeur )
    ` EXTRAIRE PARAGRAPHES ( Texte à étudier ; -> Tableau de paragraphes )

C_TEXTE ($1)
C_POINTEUR ($2)

$vlElem:=Taille tableau($2->)
Repeter
   $vlElem:=$vlElem+1
   INSERER LIGNES($2->;$vlElem)
   $vlPos:=Position(Caractere(Retour chariot);$1)
   Si ($vlPos>0)
      $2->{$vlElem}:=Sous chaine($1;1;$vlPos-1)
      $1:=Sous chaine($1;$vlPos+1)
   Sinon
      $2->{$vlElem}:=$1
   Fin de si
Jusque ($1="")
```

Référence

Position.

Longueur (chaîne) → Numérique

Paramètre	Type		Description
chaîne	Alpha	→	Chaîne dont vous voulez connaître la longueur
Résultat	Numérique	←	Nombre de caractères de chaîne

Description

Longueur vous permet d'obtenir la longueur de chaîne. Longueur retourne le nombre de caractères alphanumériques contenus dans chaîne.

Note : Le test Si(vTexte="") équivaut au test Si(Longueur(vTexte)=0).

Exemple

L'exemple suivant illustre l'utilisation de Longueur. Les valeurs retournées sont assignées à la variable vRésultat. Les commentaires fournissent la valeur de vRésultat :

```
⇒ vRésultat := Longueur ("Topaze") ` vRésultat prend la valeur 6
⇒ vRésultat := Longueur ("Citoyen") ` vRésultat prend la valeur 7
```

Code ascii (caractère) → Numérique

Paramètre	Type		Description
caractère	Alpha	→	Caractère dont vous voulez obtenir le code ASCII
Résultat	Numérique	←	Code ASCII de caractère

Description

La commande Code ascii retourne le code ASCII de caractère.

Si la chaîne caractère comporte plus d'un caractère, Code ascii retourne uniquement le code du premier caractère.

La fonction Caractere est l'inverse de Code ascii. Elle retourne le caractère désigné par un code ASCII.

Important : Dans 4D, toutes les valeurs de texte, champs ou variables, utilisent la table ASCII de MacOS, sur les plates-formes Macintosh et Windows — si aucune conversion vers une autre table ASCII n'a été effectuée. Pour plus d'informations sur ce point, reportez-vous à la section Codes ASCII.

Exemples

(1) Les caractères majuscules et minuscules ne sont pas différenciés lors d'une comparaison ou d'une recherche. Vous pouvez utiliser la fonction Code ascii si vous souhaitez établir une distinction entre les caractères majuscules et les minuscules.
En effet, cette ligne retourne VRAI :

```
("A" = "a")
```

En revanche, cette ligne retourne FAUX :

```
⇒ (Code ascii ("A") = Code ascii ("a"))
```

(2) L'exemple suivant retourne la valeur ASCII du premier caractère de la chaîne "ABC" :

```
RécupAsc := Code ascii ("ABC") ` RécupAsc prend la valeur 65, le code ASCII de A
```

(3) Le code suivant :

```
Boucle($vlCar;1;Longueur(vtText))
   Au cas ou
      : (vtText≤$vlCar≥=Caractere(Retour chariot))
         ` Faire quelque chose
      : (vtText≤$vlCar≥=Caractere(Tab))
         ` Faire autre chose
      : (...)
         ` ...
   Fin de cas
Fin de boucle
```

... lorsqu'il est utilisé de nombreuses fois avec des textes de taille importante, s'exécutera plus vite, une fois compilé, s'il est écrit ainsi :

```
   Boucle($vlCar;1;Longueur(vtText))
⇒     $vlAscii:=Code ascii(vtText≤$vlCar≥)
      Au cas ou
         : ($vlAscii=Retour chariot)
            ` Faire quelque chose
         : ($vlAscii=Tab)
            ` Faire autre chose
         : (...)
            ` ...
      Fin de cas
   Fin de boucle
```

... et ce, pour deux raisons principales : il ne référence un caractère qu'une seule fois par itération, et compare des entiers longs et non des chaînes de caractères lorsqu'il teste la présence de retours chariot et de tabulations. Nous vous conseillons d'employer cette technique lorsque vous travaillez avec des caractères ASCII standard tels que des Retours chariot et des Tabulations.

Référence

Caractere, Codes ASCII, Symboles d'indice de chaîne.

Caractere (codeASCII) → Chaîne

Paramètre	Type		Description
codeASCII	Numérique	→	Code ASCII de 0 à 255
Résultat	Chaîne	←	Caractère représenté par codeASCII

Description

La fonction Caractere retourne le caractère dont le code ASCII est codeASCII.

Astuce : La fonction Caractere est généralement utilisée pour insérer des caractères qui ne peuvent être saisis au clavier ou des caractères de contrôle dans l'éditeur de méthodes.

Important : Dans 4D, toutes les valeurs de texte, champs ou variables, utilisent la table ASCII de MacOS, sur les plates-formes Macintosh et Windows — si aucune conversion vers une autre table ASCII n'a été effectuée. Pour plus d'informations sur ce point, reportez-vous à la section Codes ASCII.

Exemple

L'exemple suivant utilise la fonction Caractere pour insérer un retour chariot dans une boîte de dialogue d'alerte afin de séparer deux lignes d'information :

```
⇒ ALERTE ("Employés : "+Chaine(Enregistrements dans table([Employés]))
                                +Caractere (13)+"Cliquez sur OK pour continuer.")
```

Référence

Code ascii, Codes ASCII, Symboles d'indice de chaîne.

Introduction

Les symboles d'indice de chaîne sont les suivants :

```
[[...]] sous Windows
≤...≥ ou [[...]] sous MacOS
```

Ces symboles sont utilisés pour désigner un caractère particulier dans une chaîne. Cette syntaxe vous permet de référencer un caractère dans un champ ou une variable de type Alpha ou Texte.

Note : Sous MacOS, vous obtenez les deux premiers symboles en appuyant sur les touches **Option**+**<** et **Option**+**>**.

Lorsque les symboles d'indice de chaîne sont placés à gauche de l'opérateur d'affectation (:=), un caractère est affecté à la position référencée dans la chaîne. Par exemple, en postulant que la chaîne vsNom n'est pas une chaîne vide, le code suivant passe le premier caractère de la chaîne vsNom en majuscule :

```
    Si (vsNom#"")
⇒       vsNom[[1]]:=Majusc(vsNom[[1]])
    Fin de si
```

Lorsque les symboles d'indice de chaîne apparaissent dans une expression, ils retournent le caractère auquel ils font référence sous la forme d'une chaîne d'un caractère. En voici un exemple :

```
    ` L'exemple suivant teste si le dernier caractère de vtText est le caractère "@"
Si (vtText # "")
    Si (Code ascii(Sous chaine(vtText;Longueur(vtText);1))=Arobase)
        ` ...
    Fin de si
Fin de si
```

```
      ` En utilisant la syntaxe des caractères d'indice de chaîne, vous écririez,
      ` plus simplement :
   Si (vtText # "")
⇒     Si (Code ascii(vtText[[Longueur(vtText)]]=Arobase)
         ` ...
      Fin de si
   Fin de si
```

Note avancée sur la référence à des caractères invalides

Lorsque vous utilisez les symboles d'indice de chaîne, **il est de votre responsabilité** de vous référer à des caractères existant dans la chaîne, de la même manière que pour les éléments d'un tableau. Si, par exemple, vous référencez le 20e caractère d'une chaîne, cette chaîne doit contenir au moins 20 caractères.

• Ne pas respecter cette condition en mode interprété n'est pas signalé comme une erreur par 4D.
• Ne pas respecter cette condition en mode compilé (sans options) peut entraîner une "corruption" de la mémoire, si, par exemple, vous écrivez un caractère au-delà de la fin d'une chaîne ou d'un texte.
• Ne pas respecter cette condition en mode compilé est signalé lorsque le contrôle d'exécution est activé. Si, par exemple, vous exécutez le code suivant :

```
   ` Ne pas faire ça !
vsToutTexte:=""
vsToutTexte[[1]]:="A"
```

L'alerte suivante s'affichera :

Exemples

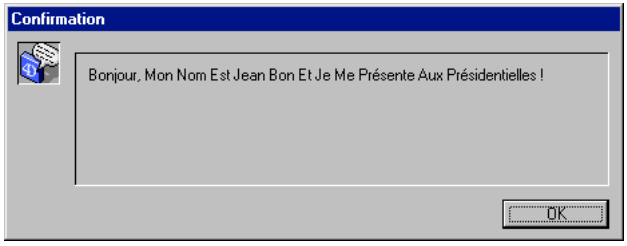
La méthode projet suivante ajoute une lettre capitale à tous les mots du texte passé en paramètre et retourne le texte modifié :

```
` Méthode projet de passage en capitale
` PasserEnCap ( Texte ) -> Texte
` PasserEnCap ( Texte source ) -> Texte avec des lettres capitales

$0:=$1
$vlLen:=Longueur($0)
Si ($vlLen>0)
   $0[[1]]:=Majusc($0[[1]])
   Boucle ($vlChar;1;$vlLen-1)
      Si (Position($0[[$vlChar]];" !&()-{}:;<>?/,.=+*")>0)
         $0[[$vlChar+1]]:=Majusc($0[[$vlChar+1]])
      Fin de si
   Fin de boucle
Fin de si
```

Une fois cette méthode placée dans la base, la ligne :

```
ALERTE(PasserEnCap ("Bonjour, mon nom est Jean Bon et je me présente aux
                                                    présidentielles !"))
```

... affiche l'alerte suivante :



Référence

Caractere, Code ascii, Codes ASCII.

Majusc (chaîne) → Alpha

Paramètre	Type		Description
chaîne	Alpha	→	Chaîne à convertir en majuscules
Résultat	Alpha	←	chaîne en majuscules

Description

Majusc retourne une chaîne de caractères égale à chaîne dont tous les caractères alphabétiques ont été convertis en majuscules.

Exemples

Reportez-vous à l'exemple de Minusc.

Référence

Minusc.

Minusc (chaîne) → Alpha

Paramètre	Type		Description
chaîne	Alpha	→	Chaîne à convertir en minuscules
Résultat	Alpha	←	chaîne en minuscules

Description

Minusc retourne une chaîne de caractères égale à chaîne dont tous les caractères alphabétiques ont été convertis en minuscules.

Exemple

L'exemple suivant est une méthode projet qui met en majuscule (capitale) le premier caractère de la chaîne ou du texte qui lui est passé(e). Par exemple, Nom := Capitale ("jean") donnerait à Nom la valeur "Jean" :

```
` Méthode projet Capitale
` Capitale ( Chaîne ) -> Chaîne
` Capitale ( Tout texte ou chaîne ) -> texte avec une lettre capitale

⇒ $0:=Minusc($1)
  Si (Longueur($0)>0)
     $0≤1≥:=Majusc($0≤1≥)
  Fin de si
```

Référence

Majusc.

Remplacer caracteres (source; nouveau; position) → Alpha

Paramètre	Type		Description
source	Alpha	→	Chaîne de départ
nouveau	Alpha	→	Nouveaux caractères
position	Numérique	→	Position de départ du remplacement
Résultat	Alpha	←	Chaîne résultante

Description

Remplacer caracteres retourne une chaîne résultant du remplacement des caractères, dans la chaîne source, à partir de position, par la chaîne nouveau.

Si nouveau est une chaîne vide (""), Remplacer caracteres retourne source inchangé. Remplacer caracteres retourne toujours une chaîne de la même longueur que source. Si position est inférieur ou supérieur à la longueur de source, Remplacer caracteres retourne source.

La fonction Remplacer caracteres est différente de Inserer chaine car elle remplace des caractères au lieu de les insérer.

Exemples

L'exemple suivant illustre l'utilisation de Remplacer caracteres. Les résultats sont affectés à la variable vRésultat.

```
⇒ vRésultat := Remplacer caracteres ("Acme"; "CME"; 2)  ` vRésultat est égal à "ACME"
   ` vRésultat est égal à "décembre"
⇒ vRésultat := Remplacer caracteres ("novembre";"déc"; 1)
```

Référence

Inserer chaine, Remplacer chaine, Supprimer chaine.

Inserer chaine (source; insertion; position) → Alpha

Paramètre	Type		Description
source	Alpha	→	Chaîne dans laquelle effectuer l'insertion
insertion	Alpha	→	Chaîne à insérer dans source
position	Numérique	→	Position de l'insertion
Résultat	Alpha	←	Chaîne résultante

Description

Inserer chaine insère la chaîne de caractères alphanumériques insertion dans la chaîne source à partir de position et retourne la chaîne de caractères résultante. La chaîne insertion est placée avant le caractère désigné par position.

Si insertion est une chaîne vide (""), Inserer chaine retourne source inchangé.

Si position est supérieur à la longueur de source, insertion est ajouté à la fin de source. Si position est inférieur à un (1), insertion est inséré au début de source.

Inserer chaine est différent de Remplacer caracteres puisque cette fonction insère des caractères au lieu de les remplacer.

Exemples

L'exemple suivant illustre l'utilisation de Inserer chaine. Les résultats sont affectés à la variable vRésultat.

```
⇒ vRésultat := Inserer chaine ("L'arbre"; " vert"; 8)  ` vRésultat est égal à "L'arbre vert"
⇒ vRésultat := Inserer chaine ("Tale"; "b"; 3)  ` vRésultat est égal à "Table"
⇒ vRésultat := Inserer chaine ("Indention"; "ta"; 6)  ` vRésultat est égal à "Indentation"
```

Référence

Remplacer caracteres, Remplacer chaine, Supprimer chaine.

Supprimer chaine (source; position; nombreCar) → Alpha

Paramètre	Type		Description
source	Alpha	→	Chaîne de départ
position	Numérique	→	Premier caractère à supprimer
nombreCar	Numérique	→	Nombre de caractères à supprimer
Résultat	Alpha	←	Chaîne résultante

Description

Supprimer chaine supprime nombreCar dans source à partir de position et retourne la chaîne résultante.

Supprimer chaine retourne la même chaîne que source dans les cas suivants :
• source est une chaîne vide,
• position est supérieur à la longueur de source,
• nombreCar est égal à zéro (0).

Si position est inférieur à un (1), les caractères sont supprimés à partir du début de la chaîne.

Si position + nombreCar est supérieur ou égal à la longueur de source, les caractères sont supprimés à partir de position jusqu'à la fin de source.

Exemples

L'exemple suivant illustre l'utilisation de Supprimer chaine. Les résultats sont affectés à la variable vRésultat.

```
⇒ vRésultat := Supprimer chaine ("Lamborghini"; 6; 6)  ` vRésultat est égal à "Lambo"
⇒ vRésultat := Supprimer chaine ("Indentation"; 6; 2)  ` vRésultat est égal à "Indention"
   ` vRésultat est égal aux deux premiers caractères de var
⇒ vRésultat := Supprimer chaine (var; 3; 32000)
```

Référence

Inserer chaine, Remplacer caracteres, Remplacer chaine.

Remplacer chaine (source; obsolète; nouveau{; remplacements}) → Alpha

Paramètre	Type		Description
source	Alpha	→	Chaîne de départ
obsolète	Alpha	→	Caractère(s) à remplacer
nouveau	Alpha	→	Chaîne de remplacement (si chaîne vide, toutes les occurrences sont effacées)
remplacements	Numérique	→	Nombre de remplacements à effectuer
Résultat	Alpha	←	Chaîne résultante

Description

Remplacer chaine retourne une chaîne de caractères résultant du remplacement dans source de obsolète par nouveau.

Si nouveau est une chaîne vide (""), Remplacer chaine supprime chaque occurrence de obsolète dans source.

Si remplacements est spécifié, Remplacer chaine ne remplace que le nombre d'occurrences de obsolète spécifié, à partir du premier caractère de source. Si remplacements est omis, toutes les occurrences de obsolète sont remplacées.

Si obsolète est une chaîne vide, Remplacer chaine retourne source inchangé.

Exemples

(1) L'exemple suivant illustre l'utilisation de Remplacer chaine. Les résultats sont affectés à la variable vRésultat. Les commentaires fournissent la valeur de la variable :

```
⇒ vRésultat := Remplacer chaine ("Ville"; "ll"; "d")        ` vRésultat est égal à "Vide"
⇒ vRésultat := Remplacer chaine ("Table"; "b"; "")          ` vRésultat est égal à "Tale"
   ` Remplacer toutes les tabulations par des virgules
⇒ vRésultat := Remplacer chaine (var; Caractere (9); ",")
```

(2) L'exemple suivant élimine les retours chariot et les tabulations du texte contenu dans la variable vRésultat :

```
⇒ vRésultat := Remplacer chaine (Remplacer chaine(vRésultat;Caractere(13);"");
                                                    Caractere(9);"")
```

Référence

Inserer chaine, Remplacer caracteres, Supprimer chaine.

Mac vers Windows (texte) → Chaîne

Paramètre	Type		Description
texte	Chaîne	→	Texte exprimé en ASCII MacOS
Résultat	Chaîne	←	Texte exprimé en ANSI Windows

Description

Mac vers Windows retourne un texte exprimé avec la table ANSI Windows équivalent au texte passé dans texte, exprimé avec la table ASCII MacOS.

Cette commande attend un paramètre de type Texte exprimé en ASCII MacOS.

En général, vous n'avez pas besoin d'utiliser cette commande. Dans 4D, chaque valeur, champ ou variable de texte est encodée sur la base de la table ASCII MacOS sous Macintosh et Windows, dans la mesure où vous ne les avez pas convertis à l'aide d'un filtre ASCII. Pour plus d'informations sur ce point, référez-vous à la section Codes ASCII.

Sous Windows, vous n'avez pas besoin de vous préoccuper de la conversion des codes ASCII. Lorsque vous copiez ou collez du texte entre 4D et Windows ou lorsque vous importez/exportez des données, 4D effectue automatiquement ces conversions. Cependant, lorsque vous utilisez des commandes de lecture/écriture telles que ENVOYER PAQUET ou RECEVOIR PAQUET, vous devez explicitement effectuer des conversions ASCII. C'est, en fait, le principal rôle de cette commande (voir exemple ci-dessous).

Exemple

Sous Windows, lorsque vous écrivez des caractères dans un document à l'aide de ENVOYER PAQUET, et si vous n'utilisez pas de filtre ASCII d'exportation pour convertir les caractères MacOS vers Windows (cf. la commande UTILISER FILTRE), il vous faut convertir vous-même le texte de MacOS vers Windows. Vous pouvez le faire de la manière suivante :

```
      ` ...
⇒     ENVOYER PAQUET ($vhDocRef;Mac vers Windows(vtTexte))
      ` ...
```

Référence

Codes ASCII, ENVOYER PAQUET, UTILISER FILTRE, Windows vers Mac.

Windows vers Mac (texte) → Chaîne

Paramètre	Type		Description
texte	Chaîne	→	Texte en ANSI Windows
Résultat	Chaîne	←	Texte en ASCII MacOS

Description

Windows vers Mac retourne un texte exprimé avec la table ASCII MacOS équivalent au texte passé dans texte, exprimé avec la table ANSI Windows.

Cette commande attend un paramètre de type Texte exprimé en ANSI Windows.

Normalement, vous n'avez pas besoin d'utiliser cette commande. Dans 4D, chaque valeur, champ ou variable de texte est encodée sur la base de la table ASCII MacOS sous Macintosh et Windows, dans la mesure où vous ne les avez pas convertis à l'aide d'un filtre ASCII. Pour plus d'informations sur ce point, référez-vous à la section Codes ASCII.

Sous Windows, vous n'avez pas besoin de vous préoccuper de la conversion des codes ASCII. Lorsque vous copiez ou collez du texte entre 4D et Windows ou lorsque vous importez/exportez des données, 4D effectue automatiquement ces conversions. Cependant, lorsque vous utilisez des commandes de lecture/écriture telles que ENVOYER PAQUET ou RECEVOIR PAQUET, vous devez explicitement effectuer des conversions ASCII. C'est, en fait, le principal rôle de cette commande (reportez-vous à l'exemple ci-dessous).

Exemple

Sous Windows, lorsque vous lisez des caractères d'un document à l'aide de RECEVOIR PAQUET, si vous n'utilisez pas de filtre ASCII d'importation pour convertir les caractères Windows vers MacOS (cf. la commande UTILISER FILTRE), il vous faut convertir vous-même le texte de Windows vers MacOS. Vous pouvez le faire de la manière suivante :

```
      ` ...
      RECEVOIR PAQUET ($vhDocRef;vtTexte;16*1024)
⇒     vtTexte:=Windows vers Mac(vtTexte)
      ` ...
```

Référence

Codes ASCII, Mac vers Windows, RECEVOIR PAQUET, UTILISER FILTRE.

Mac vers ISO (texte) → Chaîne

Paramètre	Type		Description
texte	Chaîne	→	Texte en ASCII MacOS
Résultat	Chaîne	←	Texte en ISO Latin-1

Description

Mac vers ISO retourne un texte exprimé à l'aide de la table de caractères ISO Latin-1 équivalent au texte passé dans texte, exprimé à l'aide de la table ASCII MacOS.

Cette commande attend un paramètre de type texte exprimé en ASCII MacOS.

Vous n'aurez généralement pas besoin d'utiliser cette commande. 4D convertit, dans les deux sens, les caractères reçus et envoyés par le browser Web. En résultat, les valeurs textes que vous manipulez, à l'intérieur d'un process de connexion Web, sont toutes exprimées à l'aide de la table ASCII MacOS.

Dans 4D, chaque valeur, champ ou variable de texte est encodée sur la base de la table ASCII MacOS sous Macintosh et Windows, dans la mesure où vous ne les avez pas convertis à l'aide d'un filtre ASCII. Pour plus d'informations sur ce point, référez-vous à la section Codes ASCII.

Sous Windows, vous n'avez pas besoin de vous préoccuper de la conversion des codes ASCII. Lorsque vous copiez ou collez du texte entre 4D et Windows ou lorsque vous importez/exportez des données, 4D effectue automatiquement ces conversions. Cependant, lorsque vous utilisez des commandes de lecture/écriture telles que ENVOYER PAQUET ou RECEVOIR PAQUET, 4D n'effectue aucune conversion de code ASCII.

Par conséquent, quelle que soit la plate-forme sur laquelle vous travaillez, si vous voulez écrire des documents HTML ISO Latin-1 sur disque, vous avez juste besoin de convertir le texte à l'aide de la fonction Mac vers ISO. C'est en fait le principal rôle de cette commande.

Sous Windows, dans ce cas, vous ne devez pas filtrer les caractères à l'aide d'un filtre d'exportation ASCII.

Note : La commande LIRE VARIABLES utilise la ressource “Conversion” (MapC), si elle existe. Pour plus d'informations sur ce point, reportez-vous à la documentation de Customizer Plus.

Exemples

(1) La ligne de code suivante convertit le texte encodé MacOS stocké dans vtTexte en texte encodé ISO-Latin 1 :

```
⇒    vtTexte:=Mac vers ISO(vtTexte)
```

(2) Lors du développement d'une application 4D Web Server, vous créez par programmation des pages HTML que vous enverrez par la suite sur Intranet ou Internet à l'aide de la commande ENVOYER FICHIER HTML. Dans certains de ces documents se trouvent des références ou des liens vers d'autres documents. La méthode projet ci-dessous calcule le chemin d'accès HTML à partir du chemin d'accès Windows ou Macintosh reçu en paramètre :

```
      ` Méthode projet Chemin HTML
      ` Chemin HTML ( Texte ) -> Texte
      ` Chemin HTML ( Chemin d'accès du gestionnaire de fichier natif )
                                                    -> Chemin d'accès HTML

   C_TEXTE($0;$1)
   C_ENTIER LONG($vlCar;$vlAscii)
   C_ALPHA(31;$vsCar)

   $0:=""
   Si (Sous Windows )
      $1:=Remplacer chaine($1;"\";"/")
   Sinon
      $1:=Remplacer chaine($1;":";"/")
   Fin de si
⇒  $1:=Mac vers ISO($1)
   Boucle ($vlCar;1;Longueur($1))
      $vlAscii:=Code ascii($1[[$vlCar]])
      Au cas ou
         : ($vlAscii>=127)
            $vsCar:="%"+Sous chaine(Chaine($vlAscii;"&$");2)
         : (Position(Caractere($vlAscii);":<>&%= "+Caractere(34))>0)
            $vsCar:="%"+Sous chaine(Chaine($vlAscii;"&$");2)
      Sinon
         $vsCar:=Caractere($vlAscii)
      Fin de cas
      $0:=$0+$vsCar
   Fin de boucle
```

Note : La méthode projet Sous Windows est listée dans la section Présentation des documents système.

Une fois cette méthode projet placée dans votre base, si vous voulez inclure une liste de liens FTP vers des documents présents dans un répertoire particulier, vous pouvez écrire par exemple :

```
   ` Variables interprocess définies, par exemple, dans la méthode base Sur ouverture
<>vsFTPURL:="ftp://123.4.56.78/Spiders/"
   ` Ici, un chemin du gestionnaire de fichier MacOS
<>vsFTPDirectory:="APS500:Spiders:"                ` ...

   ` ...
TABLEAU ALPHA(31;$asDocuments;0)
LISTE DES DOCUMENTS(...;$asDocuments)
$vlNbDocuments:=Taille tableau($asDocuments)
jsHandler:=...
Boucle ($vlDocument;1;$vlNbDocuments)
   vtHTMLCode:=vtHTMLCode+"<P><A HREF="+Caractere(34)+<>vsFTPURL
                        +Chemin HTML (Sous chaine($1+$asDocuments{$vlDocument};
                       Longueur (<>vsFTPDirectory)+1))+Caractere(34)+jsHandler+"> "
                            +$asDocuments{$vlDocument}+"</A></P>"+Caractere(13)
Fin de boucle
   ` ...
```

Référence

Codes ASCII, ENVOYER FICHIER HTML, ENVOYER PAQUET, ISO vers Mac, UTILISER FILTRE.

ISO vers Mac (texte) → Chaîne

Paramètre	Type		Description
texte	Chaîne	→	Texte en ISO Latin-1
Résultat	Chaîne	←	Texte en ASCII MacOS

Description

ISO vers Mac retourne un texte, exprimé à l'aide de la table ASCII MacOS, équivalent au texte passé dans texte, exprimé à l'aide de la table ISO Latin-1.

Cette commande attend un paramètre de type Texte exprimé en ISO Latin-1.

Vous n'aurez généralement pas besoin d'utiliser cette commande. 4D convertit, dans les deux sens, les caractères reçus et envoyés par les browsers Web. Comme résultat, les valeurs textes que vous gérez, dans un process de connexion Web, sont exprimées à l'aide du filtre MacOS ASCII.

Dans 4D, tous les valeurs, champs ou variables de texte que vous n'avez pas convertis à l'aide d'un autre filtre ASCII sont encodés MacOS sous Macintosh et Windows. Pour de plus amples informations sur ce sujet, référez-vous à la section Codes ASCII.

Sous Windows, vous n'avez pas besoin de vous soucier de la conversion des codes ASCII. Lorsque vous copiez ou collez du texte entre 4D et Windows ou lorsque vous importez/exportez des données, 4D effectue automatiquement ces conversions. Cependant, lorsque vous utilisez des commandes lecture/écriture comme ENVOYER PAQUET ou RECEVOIR PAQUET, il faut faire les conversions ASCII.

Par conséquent, quelle que soit la plate-forme sur laquelle vous travaillez, si vous voulez lire des documents HTML ISO Latin-1 stockés sur disque à l'aide de RECEPTION PAQUET, vous avez juste besoin de convertir le texte à l'aide de ISO vers Mac. C'est en fait le principal rôle de cette fonction.

Sous Windows, dans ce cas, vous ne devez pas filtrer les caractères à l'aide d'un filtre d'import ASCII.

Note : La commande ISO vers Mac utilise la ressource “Conversion” (MapC), si elle existe. Pour plus d'informations sur ce point, reportez-vous à la documentation de Customizer Plus.

Exemple

La ligne suivante convertit le texte encodé ISO Latin-1 stocké dans vtTexte en un texte encodé MacOS :

```
      ` Lire du texte d'un document HTML ISO Latin-1
   RECEVOIR PAQUET ($vhDocRef;vtTexte;16*1024)
⇒  vtTexte:=ISO vers Mac(vtTexte)
```

Référence

Codes ASCII, Mac vers ISO, RECEVOIR PAQUET, UTILISER FILTRE.

6

Commandes obsolètes

CHERCHER SUR CLE

Cette commande est conservée dans 4e Dimension pour des raisons de compatibilité avec les versions précédentes du programme. Pour toute nouvelle programmation, utilisez la commande CHERCHER.

ATTENTION : Cette commande disparaîtra dans les prochaines versions de 4D. Nous vous conseillons de ne plus l'utiliser.

TRIER SUR INDEX

Cette commande est conservée pour des raisons de compatibilité avec les versions précédentes de 4e Dimension. Elle est avantageusement remplacée par la commande TRIER.

ATTENTION : N'utilisez plus cette commande, elle disparaîtra dans les prochaines versions de 4D.

7

Communications

REGLER SERIE (port | opération{; param | document})

Paramètre	Type		Description
port	opération	Numérique	→	Numéro de port série ou opération à effectuer sur document
param	document	Num	Alpha	→	Paramètres de communication ou Nom du document

Description

La commande REGLER SERIE permet d'ouvrir un port série ou un document. Vous ne pouvez ouvrir qu'un port série ou un document à la fois avec cette commande.

Note historique : A l'origine, REGLER SERIE a été la première commande 4D permettant de travailler avec les ports série et des documents sur disque. Depuis, de nouvelles commandes ont été ajoutées. Aujourd'hui, vous pouvez généralement travailler avec des documents sur disque à l'aide des commandes Ouvrir document, Creer document et Ajouter a document, puis lire et écrire des caractères dans les documents avec RECEVOIR PAQUET et ENVOYER PAQUET (ces deux commandes fonctionnent aussi avec REGLER SERIE). Cependant, si vous souhaitez utiliser les commandes ENVOYER VARIABLE, RECEVOIR VARIABLE, ENVOYER ENREGISTREMENT et RECEVOIR ENREGISTREMENT, vous devez appeler REGLER SERIE pour accéder aux documents sur disque.

La description de la commande REGLER SERIE se compose de deux sections :
• Travailler avec les ports série
• Travailler avec des documents

Travailler avec les ports série : REGLER SERIE(port;param)

La première syntaxe de REGLER SERIE ouvre un port série et définit le protocole de communication ainsi que des informations supplémentaires. Les données peuvent être envoyées par les commandes ENVOYER PAQUET, ENVOYER ENREGISTREMENT ou ENVOYER VARIABLE, et reçues par les commandes RECEVOIR BUFFER, RECEVOIR PAQUET, RECEVOIR ENREGISTREMENT ou RECEVOIR VARIABLE.

• Le premier paramètre, port, définit le port et le protocole utilisés. Vous pouvez adresser jusqu'à 99 ports série (un par un).

Le tableau suivant liste les valeurs possibles du paramètre port :

Valeurs port	Description
0	Port imprimante (Mac) ou COM2 (Windows) sans protocole
1	Port modem (Mac) ou COM1 (Windows) sans protocole
20	Port imprimante (Mac) ou COM2 (Windows) avec protocole logiciel tel que XON/XOFF
21	Port modem (Mac) ou COM1 (Windows) avec protocole logiciel tel que XON/XOFF
30	Port imprimante (Mac) ou COM2 (Windows) avec protocole matériel tel que RTS/CTS)
31	Port modem (Mac) ou COM1 (Windows) avec protocole matériel tel que RTS/CTS)
101 à 199	Communication série sans protocole*
201 à 299	Communication série avec protocole logiciel tel que XON/XOFF*
301 à 399	Communication série avec protocole matériel tel que RTS/CTS*

*** Important** : La valeur que vous passez dans port doit désigner un port série "logique" reconnu par votre système d'exploitation. Par exemple, pour que vous puissiez utiliser les valeurs 101, 203 et 325, les ports série COM1, COM3 et COM25 doivent avoir été correctement configurés.

Note sur les ports série

En standard, les systèmes MacOS et Windows reconnaissent deux ports série logiques : sous MacOS, le port modem et le port imprimante ; sous Windows, les ports COM1 et COM2. Toutefois, des ports série supplémentaires peuvent être ajoutés, par l'intermédiaire de cartes d'extension. 4e Dimension n'adressait à l'origine que les deux ports série standard, et a intégré par la suite la gestion des ports série supplémentaires. Pour des raisons de compatibilité, les deux systèmes d'adressage ont été conservés.

- Si vous souhaitez adresser uniquement un port série standard (imprimante/COM2 ou modem/COM1), vous pouvez passer dans le paramètre port soit une des valeurs 0, 1, 20, 21, 30 et 31 (correspondant à l'ancien mode de fonctionnement de 4D), soit une valeur > 100 (cf. ci-dessous).
- Si vous souhaitez adresser des ports série "étendus", vous devez passer dans port (pour adresser le N^{ième} port série) la valeur N+100, augmentée éventuellement de 100 ou de 200, si vous voulez utiliser respectivement un protocole logiciel ou matériel.

Exemples :

(1) Vous souhaitez utiliser le port imprimante/COM2 sans protocole, vous pouvez utiliser l'une des syntaxes suivantes :

⇒ **REGLER SERIE** (0;param)
ou
⇒ **REGLER SERIE** (102;param)

(2) Vous souhaitez utiliser le port modem/COM1 avec le protocole XON/XOFF, vous pouvez utiliser l'une des syntaxes suivantes :

⇒ **REGLER SERIE** (21;param)
ou
⇒ **REGLER SERIE** (201;param)

(3) Vous souhaitez utiliser le port COM25 avec le protocole RTS/CTS :

⇒ **REGLER SERIE** (325;param)

• Le paramètre param permet de fixer la vitesse, le nombre de bits de données, le nombre de bits de stop et la parité. La valeur de param se calcule en additionnant les valeurs de vitesse, de bits de données, de bits de stop et de parité, telles que définies dans le tableau ci-dessous. Par exemple, pour paramétrer la communication à 1200 bauds, 8 bits de données, 1 bit de stop et aucune parité, passez 27742 (soit 94+3072+16384+0) dans param.

Contrôle	**Valeur param (à cumuler)**	**Fonction**
Vitesse	380	300
(en bauds)	189	600
	94	1200
	62	1800
	46	2400
	30	3600
	22	4800
	14	7200
	10	9600
	4	19200
	2	28800
	1	38400
	0	57600
	1022	115200
	1021	230400
Bits de données	0	5
	2048	6
	1024	7
	3072	8
Bits de stop	16384	1
	–32768	1,5
	–16384	2
Parité	0	Aucune
	4096	Impaire
	12288	Paire

Astuce : Les différentes valeurs numérique à cumuler et à passer dans les paramètres port et param (à l'exception des valeurs de COM1...COM99) sont disponibles en tant que **Constantes** prédéfinies dans le thème **Communications** de l'Explorateur, en mode Structure. Pour les valeurs de COM1...COM99, vous devez utiliser des valeurs numériques littérales.

Lorsque vous n'avez plus besoin d'un port série, vous devez le refermer. Pour cela, appelez de nouveau REGLER SERIE et passez-lui la valeur 11. Exemple :

⇒ **REGLER SERIE**(11) `Referme un port série préalablement ouvert

Travailler avec des documents : REGLER SERIE(opération;document)

La seconde syntaxe de la commande REGLER SERIE vous permet de créer, ouvrir ou fermer un document. A la différence des commandes du thème Documents système, REGLER SERIE ne permet d'ouvrir qu'un document à la fois. Le document peut être "lu à partir de" ou "écrit dans". Reportez-vous à la section Présentation des documents système pour plus d'informations sur ce point.

Le premier paramètre, opération, définit l'opération à effectuer avec le document désigné par document. Le tableau suivant dresse la liste les valeurs d'opération et le résultat obtenu, en fonction de la valeur de document.
La première colonne fournit les valeurs possibles du paramètre opération. La deuxième colonne fournit les valeurs possibles du paramètre document. La troisième colonne décrit le résultat obtenu. Par exemple, pour afficher un fichier de type texte dans une boîte de dialogue standard d'ouverture de document, vous pouvez écrire l'instruction suivante :

⇒ **REGLER SERIE** (13; "")

Opération	Document	Résultat
10	Chaîne	Ouvre le document dont le nom est spécifié par Chaîne. Si le document n'existe pas, il est créé et ouvert.
10	"" (chaîne vide)	Affiche la boîte de dialogue d'ouverture de fichier. Tous les types de fichiers sont présentés.
11	Aucun	Referme un fichier ouvert.
12	"" (chaîne vide)	Affiche la boîte de dialogue standard d'enregistement de fichier, permettant de créer un nouveau fichier.
13	"" (chaîne vide)	Affiche la boîte de dialogue d'ouverture de fichier. Seuls les fichiers de type Texte sont présentés.

Toutes les opérations décrites dans ce tableau modifient la variable système Document en conséquence. De plus, la variable système OK prend la valeur 1 si l'opération s'est déroulée correctement, 0 sinon.

Exemples

Reportez-vous aux exemples des commandes RECEVOIR BUFFER, FIXER TIMEOUT et RECEVOIR ENREGISTREMENT.

Référence

Ajouter a document, Creer document, ENVOYER ENREGISTREMENT, ENVOYER PAQUET, ENVOYER VARIABLE, FIXER TIMEOUT, Ouvrir document, RECEVOIR BUFFER, RECEVOIR ENREGISTREMENT, RECEVOIR PAQUET, RECEVOIR VARIABLE.

FIXER TIMEOUT (secondes)

Paramètre	Type		Description
secondes	Numérique	→	Nombre de secondes jusqu'au timeout

Description

La commande FIXER TIMEOUT vous permet de définir le temps d'attente maximum pour l'exécution d'une commande de communication série. Si la commande ne se termine pas dans le temps secondes qui lui est imparti, la communication série est annulée, l'erreur –9990 est générée, et la variable système OK prend la valeur 0. Vous pouvez intercepter cette erreur à l'aide d'une méthode installée par la commande APPELER SUR ERREUR.

Notez que le délai défini représente la durée totale permise pour que la commande s'exécute, et non le délai d'attente entre chaque caractère reçu. Pour annuler un paramétrage précédent et ne pas spécifier de temps d'attente maximum, passez 0 dans le paramètre secondes.

Les commandes de communication série affectées par ce paramétrage sont les suivantes :

- RECEVOIR PAQUET
- RECEVOIR ENREGISTREMENT
- RECEVOIR VARIABLE

Exemple

L'exemple suivant définit le port série devant recevoir des données et le timeout. Les données sont lues à l'aide de RECEVOIR PAQUET. Si les données ne sont pas bien reçues dans le temps imparti, une erreur survient :

```
    ` Ouverture du port série
REGLER SERIE (Port série MacOS; Vitesse 9600 + Bits de données 8 + Bit de stop un
                                                                + Pas de parité)
⇒   FIXER TIMEOUT (10)  ` Fixer le timeout à 10 secondes
    ` Traiter les interruptions éventuelles
APPELER SUR ERREUR ("INTERCEPTER ERREURS COMMUNICATIONS")
RECEVOIR PAQUET (vBuffer; Caractere (Retour chariot))  ` Lire jusqu'au retour chariot
Si (OK = 0)  ` Si une erreur survient
   ALERTE ("Erreur lors de la réception des données.")  ` Informer l'utilisateur
Sinon
   [Personnes]Nom := vBuffer  ` Sauvegarder les données dans un champ
Fin de si
```

Référence

APPELER SUR ERREUR, RECEVOIR BUFFER, RECEVOIR ENREGISTREMENT, RECEVOIR PAQUET, RECEVOIR VARIABLE.

UTILISER FILTRE (filtre | *{; typeFiltre})

Paramètre	Type		Description
filtre | *	Alpha | *	→	Nom du document filtre ASCII à utiliser ou * pour restaurer le filtre ASCII par défaut
typeFiltre	Numérique	→	0 = Filtre d'exportation, 1 = Filtre d'importation

Description

La commande UTILISER FILTRE a deux syntaxes. La première charge en mémoire le document de filtre ASCII filtre préalablement sauvegardé et l'utilise.

Si typeFiltre est égal à 0, le filtre est utilisé en tant que filtre d'exportation. Si typeFiltre est égal à 1, il est utilisé en tant que filtre d'importation. Si vous ne passez pas le paramètre typeFiltre, le filtre d'exportation est utilisé par défaut.

Le filtre ASCII doit avoir été préalablement créé en mode Utilisation, dans la boîte de dialogue de création de filtres. Une fois chargé en mémoire, ce filtre sera utilisé par 4e Dimension pour toutes les opérations de transfert entre la base et un document ou un port série. Cela inclut les données transférées par les commandes d'import/export ASCII, SYLK et DIF, ainsi que celles envoyées par les commandes ENVOYER PAQUET, RECEVOIR PAQUET et RECEVOIR BUFFER. Les filtres n'ont pas d'effet sur les données transférées par les commandes ENVOYER ENREGISTREMENT, ENVOYER VARIABLE, RECEVOIR ENREGISTREMENT et RECEVOIR VARIABLE.

Si vous passez une chaîne vide dans le paramètre filtre, UTILISER FILTRE affiche une boîte de dialogue standard d'ouverture de fichiers pour que l'utilisateur puisse sélectionner un filtre ASCII. Lorsque vous appelez UTILISER FILTRE, la variable système OK prend la valeur 1 si le filtre est correctement chargé, sinon elle prend la valeur 0.

La deuxième syntaxe de UTILISER FILTRE, lorsque vous passez un astérisque * au lieu du paramètre filtre, restaure le filtre ASCII par défaut. Si typeFiltre est égal à 0, le filtre d'exportation est restauré. Si typeFiltre est égal à 1, le filtre d'importation est restauré. Le filtre ASCII par défaut n'établit pas de traduction entre les caractères. Lorsque cette seconde syntaxe est utilisée, la variable système OK prend la valeur 0.

Exemple

L'exemple suivant charge en mémoire un document de filtre ASCII. Les données sont exportées, puis le filtre ASCII par défaut est restauré :

```
⇒ UTILISER FILTRE ("MactoPC"; 0) ` Charger un filtre ASCII
  ECRITURE ASCII ([Ma Table]; "Mon Texte") ` Exporter les données avec le filtre ASCII
⇒ UTILISER FILTRE (*; 0) ` Restaurer le filtre ASCII par défaut
```

Référence

ECRITURE ASCII, ECRITURE DIF, ECRITURE SYLK, ENVOYER PAQUET, LECTURE ASCII, LECTURE DIF, LECTURE SYLK, Mac vers Windows, RECEVOIR BUFFER, RECEVOIR PAQUET, Windows vers Mac.

ENVOYER PAQUET ({docRef; }paquet)

Paramètre	Type		Description
docRef	docRef	→	Référence de document ou canal courant (port série ou document)
paquet	Alpha	→	Chaîne ou texte à envoyer

Description

La commande ENVOYER PAQUET envoie paquet vers un port série ou un document. Si docRef est spécifié, le paquet est écrit dans le document référencé par docRef. Si docRef n'est pas spécifié, le paquet est envoyé vers le port série ou un document préalablement ouvert par la commande REGLER SERIE. paquet représente une simple série de données, généralement une chaîne de caractères.

Avant d'utiliser ENVOYER PAQUET, vous devez ouvrir un port série ou un document avec la commande REGLER SERIE, ou un document avec une commande de gestion des documents.

Lorsque vous envoyez un paquet vers un document, le premier ENVOYER PAQUET commence à écrire les données au début du document — à moins que ce dernier n'ait été ouvert par la fonction Ajouter a document. Puis, jusqu'à ce que le document soit refermé, chaque paquet envoyé y est écrit à la suite du précédent.

Note 4D version 6 : Ce fonctionnement est toujours valide avec un document ouvert par REGLER SERIE. Cependant, pour un document ouvert par Ouvrir document, Creer document ou Ajouter a document, vous pouvez désormais utiliser les nouvelles commandes Position dans document et CHANGER POSITION DANS DOCUMENT pour connaître et modifier la position à laquelle, dans le document, la prochaine écriture (ENVOYER PAQUET) ou lecture (RECEVOIR PAQUET) aura lieu.

Important : ENVOYER PAQUET envoie des caractères ASCII MacOS sous Windows et sous MacOS. Chacun d'entre eux est codé sur huit bits. Les caractères ASCII standard utilisent uniquement les sept bits inférieurs. De nombreux ordinateurs et périphériques n'utilisent pas le huitième bit de la même manière que Windows/MacOS. Si la chaîne à envoyer contient des caractères utilisant le huitième bit, créez un filtre ASCII pour convertir les caractères ASCII, et exécutez UTILISER FILTRE avant d'utiliser ENVOYER PAQUET. Vous pouvez aussi utiliser la fonction Mac vers Windows (voir l'exemple de cette fonction). Des protocoles tels que XON/XOFF utilisent certains codes ASCII inférieurs pour établir la communication entre les machines. Assurez-vous de ne pas envoyer de tels codes ASCII pour ne pas risquer d'interférer avec le protocole, voire de rompre la communication.

Exemple

L'exemple suivant écrit dans un document des données en provenance de champs. Les valeurs sont écrites sous forme de champs de taille fixe. Dans ce cas, si la longueur d'un champ est inférieure à la taille fixée, le champ est comblé avec des espaces (c'est-à-dire que des espaces sont ajoutés de manière à ce que le champ corresponde à la taille définie). Bien que les champs de valeurs fixes soient un moyen peu efficace de stocker des données, certains systèmes informatiques et certaines applications l'utilisent encore :

```
$Doc := Creer document ("")    ` Création d'un document
Si (OK=1)    ` Est-ce que le document a bien été créé ?
      ` Boucle pour chaque enregistrement
   Boucle ($i; 1; Enregistrements trouves ([Personnes]))
         ` Envoi du paquet créé à partir d'une chaîne de 15 espaces contenant le
         ` champ Prénom
⇒        ENVOYER PAQUET ($Doc; Remplacer caracteres(15 * Caractere(Espacement);
                                                    [Personnes]Prénom;1))
⇒        ENVOYER PAQUET ($Doc; Remplacer caracteres(15 * Caractere(Espacement);
                                                    [Personnes]Nom;1))
         ` Envoi d'un second paquet créé à partir d'un chaîne de 15 espaces contenant
         ` le champ Nom. Cela aurait pu être mis dans le premier paquet, mais est
         ` séparé pour des raisons de clarté
      ENREGISTREMENT SUIVANT([Personnes])
   Fin de boucle
      ` Envoi du code ASCII SUB, utilisé comme marqueur de fin d'enregistrement par
      ` certains ordinateurs.
⇒     ENVOYER PAQUET ($Doc; Caractere (ASCII SUB))
   FERMER DOCUMENT ($Doc)    ` Fermeture du document
Fin de si
```

Référence

CHANGER POSITION DANS DOCUMENT, Position dans document, RECEVOIR PAQUET.

RECEVOIR PAQUET ({docRef; }réceptVar; stopCar | nbCar)

Paramètre	Type		Description
docRef	docRef	→	Numéro de référence de document ou canal courant (port série ou document)
réceptVar	Variable	→	Variable devant recevoir les données
stopCar | nbCar	Alpha | Num	→	Caractère auquel stopper la réception des données ou Nombre de caractères à recevoir

Description

La commande RECEVOIR PAQUET lit des caractères depuis un port série ou un document.

Si docRef est spécifié, la commande récupère des caractères depuis un document ouvert par la fonction Ouvrir document, Creer document ou Ajouter a document. Si docRef est omis, la commande récupère des caractères depuis un port série ou un document ouvert par la commande REGLER SERIE.

Dans tous les cas, les caractères lus sont retournés dans la variable réceptVar, qui doit être une variable de type Texte ou Alpha. Si vous voulez lire un nombre prédéfini de caractères, passez ce nombre dans le paramètre nbCar. Si vous voulez lire des caractères jusqu'à ce qu'un caractère spécifié soit lu, passez ce caractère dans le paramètre stopCar (le caractère d'arrêt n'est pas retourné dans réceptVar).

Lorsque RECEVOIR PAQUET lit un document, si vous n'avez spécifié ni nbCar ni stopCar, l'exécution de la commande se terminera à la fin du document. Cependant, rappelez-vous que si une variable chaîne a une taille fixe, une variable texte quant à elle accepte jusqu'à 32000 caractères. Si RECEVOIR PAQUET doit lire des données en provenance du port série, la commande s'exécutera indéfiniment, jusqu'à ce que le délai d'attente (s'il est fixé) soit écoulé (cf. la commande FIXER TIMEOUT) ou que l'utilisateur interrompe la réception (voir ci-dessous).

Pendant l'exécution d'un RECEVOIR PAQUET, l'utilisateur peut interrompre l'opération en appuyant sur les touches **Ctrl**+**Alt**+**Maj** (sous Windows) ou **Commande**+**Option**+**Maj** (sous MacOS). Cette interruption génère une erreur –9994 que vous pouvez intercepter à l'aide d'une méthode installée par la commande APPELER SUR ERREUR. Généralement, vous devez gérer les interruptions d'une réception uniquement lors d'une communication série.

Lors de la lecture d'un document, le premier RECEVOIR PAQUET commence par lire le début du document. La lecture des paquets suivants débute au caractère situé immédiatement après le dernier caractère lu.

Note 4D version 6 : Ce fonctionnement est toujours valide avec un document ouvert par REGLER SERIE. Cependant, pour un document ouvert par Ouvrir document, Creer document ou Ajouter a document, vous pouvez désormais utiliser les nouvelles commandes Position dans document et CHANGER POSITION DANS DOCUMENT pour connaître et modifier la position à laquelle, dans le document, la prochaine écriture (ENVOYER PAQUET) ou lecture (RECEVOIR PAQUET) aura lieu.

En cas de tentative de lecture après la fin d'un document, RECEVOIR PAQUET retourne les données lues jusqu'à ce point et la variable système OK prend la valeur 1. Les RECEVOIR PAQUET suivants retourneront une chaîne vide et OK prendra la valeur zéro.

Note : Sous Windows, lorsque vous lisez des caractères d'un document à l'aide de RECEVOIR PAQUET, et si vous n'utilisez pas de filtre ASCII d'importation pour convertir les caractères Windows vers MacOS (cf. la commande UTILISER FILTRE), vous pouvez utiliser la fonction Windows vers Mac (voir l'exemple de cette fonction).

Exemples

(1) L'exemple suivant lit 20 caractères depuis un port série et les place dans la variable RécupVingt :

```
⇒ RECEVOIR PAQUET (RécupVingt; 20)
```

(2) L'exemple suivant lit des données depuis le document référencé par la variable MonDoc et les place dans la variable vData. La commande récupère les données jusqu'à ce qu'elle rencontre un retour chariot :

```
⇒ RECEVOIR PAQUET (MonDoc; vData; Caractere (Retour chariot))
```

(3) L'exemple suivant lit des données d'un document et les place dans des champs. Les données sont stockées dans des champs de longueur fixe. La méthode fait appel à une sous-routine pour éliminer les espaces superflus (situés derrière les valeurs). Le code de la sous-routine est présenté après la méthode :

```
$Doc := Ouvrir document ("";"TEXT")  ` Ouverture d'un document de type Texte
Si (OK=1)  ` Si le document est ouvert...
   Repeter  ` Boucle jusqu'à ce qu'il n'y ait plus de données
⇒     RECEVOIR PAQUET ($Doc; $Var1; 15)  ` Lecture de 15 caractères
⇒     RECEVOIR PAQUET ($Doc; $Var2; 15)  ` Même chose pour le second champ
      Si (OK = 1)  ` Si ce n'est pas la fin du document...
         CREER ENREGISTREMENT([Personnes])  ` Créer un nouvel enregistrement
         [Personnes]Prénom := Elimine ($Var1)  ` Sauvegarder le prénom
         [Personnes]Nom := Elimine ($Var2)  ` Sauvegarder le nom
         STOCKER ENREGISTREMENT([Personnes])  ` Sauvegarder l'enregistrement
      Fin de si
   Jusque (OK =0)
   FERMER DOCUMENT ($Doc)  ` Fermeture du document
Fin de si
```

Les espaces superflus derrière les valeurs sont éliminés par la méthode suivante, appelée Elimine :

```
Boucle ($i; Longueur ($1); 1; –1)  ` Boucle sur la fin de la chaîne d'où démarrer
   Si ($1≤$i≥ # " ")  ` Si ce n'est pas un espace...
      $i := –$i  ` Forcer la boucle à stopper
   Fin de si
Fin de boucle
$0 := Supprimer chaine($1; –$i;Longueur($1))  ` Suppression des espaces
```

Référence

CHANGER POSITION DANS DOCUMENT, ENVOYER PAQUET, FIXER TIMEOUT, Position dans document, RECEVOIR PAQUET.

Variables et ensembles système

Après un appel à RECEVOIR PAQUET, la variable système OK prend la valeur 1 si le paquet est reçu sans erreur. Sinon, OK prend la valeur 0.

RECEVOIR BUFFER (varRéception)

Paramètre	Type		Description
varRéception	Variable	→	Variable devant recevoir les données

Description

La commande RECEVOIR BUFFER lit les données du port série préalablement ouvert par la commande REGLER SERIE. Le port série comporte un buffer qui se remplit de caractères jusqu'à ce qu'une commande les charge. RECEVOIR BUFFER récupère les caractères présents dans le buffer, les place dans la variable varRéception puis vide le buffer. S'il n'y a pas de caractères dans le buffer, la variable varRéception est vide.

Sous Windows
Le buffer du port série sous Windows a une capacité limitée. Cela signifie que le buffer peut être saturé. Lorsqu'il est plein et que de nouveaux caractères sont reçus, ils remplacent les plus anciens caractères. Les anciens caractères sont perdus ; par conséquent, il est important que le buffer soit lu rapidement lorsque des nouvelles données sont reçues.

Sous MacOS
Le buffer du port série sous MacOS a une capacité limitée à 64 caractères. Cela signifie que le buffer est plein lorsqu'il contient 64 caractères. Lorsqu'il est saturé et que de nouveaux caractères sont reçus, ils remplacent les plus anciens caractères. Les anciens caractères sont perdus ; par conséquent, il est important que le buffer soit lu rapidement lorsque des nouvelles données sont reçues.

Note : Il existe des plug-ins 4D qui vous permettent d'augmenter la capacité du port série.

La commande RECEVOIR BUFFER est différente de RECEVOIR PAQUET dans la mesure où elle récupère tout ce qui se trouve dans le buffer et le retourne immédiatement. RECEVOIR PAQUET, pour sa part, attend de récupérer un caractère spécifique ou un certain nombre de caractères.

Pendant l'exécution d'un RECEVOIR BUFFER, l'utilisateur peut interrompre l'opération en appuyant sur les touches **Ctrl**+**Alt**+**Maj** (sous Windows) ou **Commande**+**Option**+**Maj** (sous MacOS). Cette interruption génère une erreur –9994 que vous pouvez intercepter à l'aide d'une méthode installée par la commande APPELER SUR ERREUR.

Exemple

La méthode projet ECOUTER PORT SÉRIE utilise RECEVOIR BUFFER pour récupérer du texte depuis le port série et l'accumuler dans une variable interprocess :

```
   ` ECOUTER PORT SÉRIE
   ` Ouvrir le port série
REGLER SERIE (201; Vitesse 9600+Bits de données 8+Bit de stop un+Pas de parité)
◊IP_Ecouter_Port_Série:=Vrai
Tant que (◊IP_Ecouter_Port_Série)
⇒     RECEVOIR BUFFER($vtBuffer)
      Si ((Longueur($vtBuffer)+Longueur(◊vtBuffer))>MAXLARGTEXTE)
         ◊vtBuffer:=""
      Fin de si
      ◊vtBuffer:=◊vtBuffer+$Buffer
Fin tant que
```

A ce stade, tout autre process peut lire la variable interprocess ◊vtBuffer pour exploiter les données en provenance du port série.

Pour cesser d'écouter le port série, exécutez simplement la méthode suivante :

```
   ` Fin de l'écoute du port série
◊IP_Ecouter_Port_Série:=Faux
```

Notez que l'accès à la variable interprocess ◊vtBuffer doit être protégé par un sémaphore, de manière à ce que les process n'entrent pas en conflit (reportez-vous à la description de la fonction Semaphore pour plus d'informations).

Référence

APPELER SUR ERREUR, RECEVOIR PAQUET, REGLER SERIE, Semaphore, Variables.

ENVOYER VARIABLE (variable)

Paramètre	Type		Description
variable	Variable	→	Variable à envoyer

Description

ENVOYER VARIABLE envoie variable vers le document ou le port série préalablement ouvert par la commande REGLER SERIE. La variable est envoyée dans un format interne spécial qui ne peut être relu que par la commande RECEVOIR VARIABLE. ENVOYER VARIABLE envoie la totalité de la variable (y compris son type et sa valeur).

Notes

1. Si vous envoyez une variable à un document avec cette commande, le document doit avoir été ouvert par la commande REGLER SERIE. Vous ne pouvez pas utiliser ENVOYER VARIABLE avec un document ouvert par Ouvrir document, Ajouter a document ou Creer document.
2. Cette commande ne fonctionne pas avec les variables de type tableau. Si vous voulez envoyer et recevoir des tableaux via un document ou un port série, utilisez les nouvelles Commandes du thème BLOB, apparues avec la version 6 de 4D.

Exemple

Reportez-vous à l'exemple de la commande RECEVOIR ENREGISTREMENT.

Référence

ENVOYER ENREGISTREMENT, RECEVOIR ENREGISTREMENT, RECEVOIR VARIABLE, REGLER SERIE.

RECEVOIR VARIABLE (variable)

Paramètre	Type		Description
variable	Variable	→	Variable dans laquelle recevoir une variable

Description

La commande RECEVOIR VARIABLE reçoit variable, une variable envoyée par la commande ENVOYER VARIABLE, depuis un document ou un port série préalablement ouvert par la commande REGLER SERIE.

En mode interprété, si la variable n'existe pas préalablement à l'appel de RECEVOIR VARIABLE, elle sera créée, typée et remplie en fonction de ce qui a été reçu. En mode compilé, la variable doit être du même type que celle qui est reçue. Dans les deux cas, le contenu de la variable est remplacé par celui de la variable reçue.

Notes

1. Si vous recevez une variable depuis un document avec cette commande, le document doit avoir été ouvert par la commande REGLER SERIE. Vous ne pouvez pas utiliser RECEVOIR VARIABLE avec un document ouvert par Ouvrir document, Ajouter a document ou Creer document.
2. Cette commande ne fonctionne pas avec les variables de type tableau. Si vous voulez envoyer et recevoir des tableaux via un document ou un port série, utilisez les nouvelles Commandes du thème BLOB, apparues avec la version 6 de 4D.
3. Pendant l'exécution d'un RECEVOIR VARIABLE, l'utilisateur peut interrompre l'opération en appuyant sur les touches **Ctrl**+**Alt**+**Maj** (sous Windows) ou **Commande**+**Option**+**Maj** (sous MacOS). Cette interruption génère une erreur –9994 que vous pouvez intercepter à l'aide d'une méthode installée par la commande APPELER SUR ERREUR. Généralement, vous devez gérer les interruptions d'une réception uniquement lors d'une communication série.

Exemple

Reportez-vous à l'exemple de la commande RECEVOIR ENREGISTREMENT.

Référence

APPELER SUR ERREUR, ENVOYER ENREGISTREMENT, ENVOYER VARIABLE.

Variables et ensembles système

La variable système OK vaut 1 si la variable est correctement reçue, sinon elle vaut 0.

ENVOYER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle envoyer l'enregistrement courant ou Table par défaut si ce paramètre est omis

Description

ENVOYER ENREGISTREMENT envoie l'enregistrement courant de table vers le port série ou vers un document ouvert par la commande REGLER SERIE. L'enregistrement est envoyé dans un format interne particulier ne pouvant être interprété que par la commande RECEVOIR ENREGISTREMENT. S'il n'y a pas d'enregistrement courant, ENVOYER ENREGISTREMENT ne fait rien.

L'enregistrement est envoyé en totalité, ce qui signifie que les sous-enregistrements, les images et les BLOBs stockés dans l'enregistrement sont également envoyés.

Important : Lorsque des enregistrements sont envoyés et reçus par ENVOYER ENREGISTREMENT et RECEVOIR ENREGISTREMENT, la structure de la table source et celle de la table de destination doivent être compatibles. Si ce n'est pas le cas, 4D convertira les valeurs en fonction des définitions des tables lorsque RECEVOIR ENREGISTREMENT sera exécutée.

Note : Si vous envoyez un enregistrement à un document avec cette commande, le document doit avoir été ouvert par la commande REGLER SERIE. Vous ne pouvez pas utiliser ENVOYER ENREGISTREMENT avec un document ouvert par Ouvrir document, Ajouter a document ou Creer document.

Exemple

Reportez-vous à l'exemple de la commande RECEVOIR ENREGISTREMENT.

Référence

ENVOYER VARIABLE, RECEVOIR ENREGISTREMENT, RECEVOIR VARIABLE.

RECEVOIR ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table dans laquelle recevoir l'enregistrement ou Table par défaut si ce paramètre est omis

Description

RECEVOIR ENREGISTREMENT ajoute dans table un enregistrement reçu par l'intermédiaire du port série ou d'un document ouvert par la commande REGLER SERIE. L'enregistrement doit avoir été envoyé par la commande ENVOYER ENREGISTREMENT. Lorsque vous exécutez RECEVOIR ENREGISTREMENT, un nouvel enregistrement est automatiquement créé dans table. Si l'enregistrement a été correctement reçu, vous pouvez le sauvegarder à l'aide de STOCKER ENREGISTREMENT.

L'enregistrement est reçu en totalité, ce qui signifie que tous les sous-enregistrements, images et BLOBs stockés dans l'enregistrement sont également reçus.

Important : Lorsque des enregistrements sont envoyés et reçus par ENVOYER ENREGISTREMENT et RECEVOIR ENREGISTREMENT, la structure de la table source et celle de la table de destination doivent être compatibles. Si ce n'est pas le cas, 4D convertira les valeurs en fonction des définitions des tables lorsque RECEVOIR ENREGISTREMENT sera exécutée.

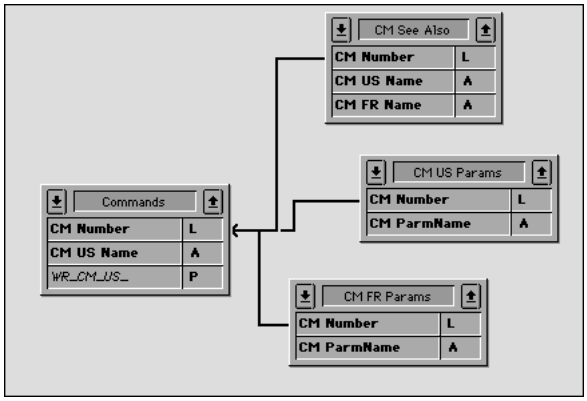
Notes

1. Si vous recevez un enregistrement provenant d'un document avec cette commande, le document doit avoir été ouvert par la commande REGLER SERIE. Vous ne pouvez pas utiliser RECEVOIR ENREGISTREMENT avec un document ouvert par Ouvrir document, Ajouter a document ou Creer document.
2. Pendant l'exécution d'un RECEVOIR ENREGISTREMENT, l'utilisateur peut interrompre l'opération en appuyant sur les touches **Ctrl**+**Alt**+**Maj** (sous Windows) ou **Commande**+**Option**+**Maj** (sous MacOS). Cette interruption génère une erreur –9994 que vous pouvez intercepter à l'aide d'une méthode installée par la commande APPELER SUR ERREUR. Généralement, vous devez gérer les interruptions d'une réception uniquement lors d'une communication série.

Exemple

L'utilisation combinée de ENVOYER VARIABLE, ENVOYER ENREGISTREMENT, RECEVOIR VARIABLE et RECEVOIR ENREGISTREMENT est idéale pour archiver des données ou échanger des données entre des bases monopostes identiques utilisées à différents endroits. Certes, vous pouvez échanger des données entre des bases 4D à l'aide des commandes d'import/export telles que ECRITURE ASCII et LECTURE ASCII. Cependant, si vos données contiennent des images, des sous-tables et/ou des tables liées, l'utilisation de ENVOYER ENREGISTREMENT et RECEVOIR ENREGISTREMENT s'avère, de loin, plus pratique.

Par exemple, la documentation que vous êtes en train de lire a été créée à l'aide de 4D et 4D Write. Comme plusieurs rédacteurs basés dans différents pays travaillaient sur ce projet, nous avions besoin d'un système simple pour échanger les données entre les différentes bases. Voici une vue simplifiée de la structure de la base :



La table [Commands] contient la description de chaque commande ou section. Les tables [CM US Params] et [CM FR Params] contiennent repectivement les paramètres de chaque commande en anglais et en français. La table [CM See Also] contient les commandes indiquées en tant que Références pour chaque commande ou section. L'échange de la documentation entre les bases consiste donc à envoyer les enregistrements de [Commands] ainsi que leurs enregistrements liés. Pour cela, nous utilisons ENVOYER ENREGISTREMENT et RECEVOIR ENREGISTREMENT. De plus, nous utilisons ENVOYER VARIABLE et RECEVOIR VARIABLE pour "cocher" les enregistrements importés/exportés.

Voici la méthode projet (simplifiée) d'export de la documentation :

```
      ` Méthode projet CM_EXPORT_SEL
      ` Cette méthode fonctionne avec la sélection courante de la table [Commands]

⇒     REGLER SERIE(12;"") ` Laissons l'utilisateur créer et ouvrir un document série
      Si (OK=1)
         ` Marquons le document avec une variable décrivant son contenu
         ` Note: la variable process BUILD_LANG indique si des données US (anglaises)
         ` ou FR (françaises) sont envoyées
         $vsTag:="4DV6COMMAND"+BUILD_LANG
⇒        ENVOYER VARIABLE($vsTag)
         ` Envoyer une variable indiquant combien de [Commands] sont exportées
         $vlNbCmd:=Enregistrements trouves([Commands])
⇒        ENVOYER VARIABLE($vlNbCmd)
         DEBUT SELECTION([Commands])
            ` Pour chaque commande
         Boucle ($vlCmd;1;$vlNbCmd)
            ` Envoyer l'enregistrement [Commands]
⇒           ENVOYER ENREGISTREMENT([Commands])
            ` Sélection de tous les enregistrements liés
            LIEN RETOUR([Commands])
            ` En fonction de la langue, envoyer une variable indiquant
            ` le nombre de paramètres qui va suivre
            Au cas ou
               : (BUILD_LANG="US")
                  $vlNbParm:=Enregistrements trouves([CM US Params])
               : (BUILD_LANG="FR")
                  $vlNbParm:=Enregistrements trouves([CM FR Params])
            Fin de cas
⇒           ENVOYER VARIABLE($vlNbParm)
            ` Envoyer les enregistrements des paramètres (s'il y en a)
            Boucle ($vlParm;1;$vlNbParm)
               Au cas ou
                  : (BUILD_LANG="US")
⇒                    ENVOYER ENREGISTREMENT([CM US Params])
                     ENREGISTREMENT SUIVANT([CM US Params])
                  : (BUILD_LANG="FR")
⇒                    ENVOYER ENREGISTREMENT([CM FR Params])
                     ENREGISTREMENT SUIVANT([CM FR Params])
               Fin de cas
            Fin de boucle
            ` Envoyer une variable indiquant combien de “Références” vont suivre
            $vlNbSee:=Enregistrements trouves([CM See Also])
```

```
⇒            ENVOYER VARIABLE($vlNbSee)
                ` Envoyer les enregistrements [See Also] (s'il y en a)
             Boucle ($vlSee;1;$vlNbSee)
⇒               ENVOYER ENREGISTREMENT([CM See Also])
                ENREGISTREMENT SUIVANT([CM See Also])
             Fin de boucle
                ` Aller à l'enregistrement [Commands] suivant et continuer l'export
             ENREGISTREMENT SUIVANT([Commands])
          Fin de boucle
⇒         REGLER SERIE(11) ` Fermer le document
       Fin de si
```

Voici la méthode projet (simplifiée) d'import de la documentation :

```
          ` Méthode projet CM_IMPORT_SEL

⇒       REGLER SERIE(10;"")   ` Laissons l'utilisateur ouvrir un document existant
        Si (OK=1) ` Si un document a été ouvert
⇒          RECEVOIR VARIABLE($vsTag)   ` Essayons de recevoir la variable marqueur attendue
           Si ($vsTag="4DV6COMMAND@")   ` Avons-nous le bon marqueur ?
                ` Extrayons la langue du marqueur
              $CurLang:=Sous chaine($vsTag;Longueur($vsTag)-1)
              Si (($CurLang="US") | ($CurLang="FR"))   ` Avons-nous reçu un langage valide ?
                    ` Combien de commandes dans ce document ?
⇒                RECEVOIR VARIABLE($vlNbCmd)
                 Si ($vlNbCmd>0)   ` S'il en existe une au moins
                       ` Pour chaque enregistrement [Commands] archivé
                    Boucle ($vlCmd;1;$vlNbCmd)
                          ` Réception de l'enregistrement
⇒                      RECEVOIR ENREGISTREMENT([Commands])
                          ` Appelons une sous-routine qui sauvegarde le nouvel
                          ` enregistrement ou le copie dans un enregistrement existant
                       CM_IMP_CMD ($CurLang)
                          ` Réception du nombre de paramètres (s'il y en a)
⇒                      RECEVOIR VARIABLE($vlNbParm)
                       Si ($vlNbParm>=0)
                             ` Appelons une sous-routine qui appelle RECEVOIR
                             ` ENREGISTREMENT puis stocke les nouveaux enregistrements ou
                             ` les copie dans des enregistrements existants
                          CM_IMP_PARM ($vlNbParm;$CurLang)
                       Fin de si
```

```
                        ` Réception du nombre de “Références” (s'il y en a)
⇒                       RECEVOIR VARIABLE($vlNbSee)
                        Si ($vlNbSee>0)
                              ` Appelons une sous-routine qui appelle RECEVOIR
                              ` ENREGISTREMENT puis stocke les nouveaux enregistrements
                              ` ou les copie dans des enregistrements existants
                           CM_IMP_SEEA ($vlNbSee;$CurLang)
                        Fin de si
                     Fin de boucle
                  Sinon
                     ALERTE("Le nombre de commandes dans ce document d'export est
                                                                              invalide.")
                  Fin de si
               Sinon
                  ALERTE("Le langage de ce document d'export est inconnu.")
               Fin de si
            Sinon
               ALERTE("Ce document n'est pas un document d'export.")
            Fin de si
⇒           REGLER SERIE(11)   ` Fermer document
         Fin de si
```

Notez que nous n'avons pas testé la variable OK pendant la réception des données, ni intercepté les éventuelles erreurs. Cependant, comme nous avons stocké dans le document des variables décrivant le document lui-même, si ces variables, une fois reçues, sont correctes, la probabilité d'erreur est très faible. Si par exemple un utilisateur ouvre un mauvais document, le premier test stoppe toute l'opération.

Référence

ENVOYER ENREGISTREMENT, ENVOYER VARIABLE, RECEVOIR VARIABLE.

Variables et ensembles système

La variable système OK prend la valeur 1 si l'enregistrement est correctement reçu, sinon elle prend la valeur 0.

8

Compilateur

4D Compiler vous permet de traduire vos applications de base de données en instructions de niveau assembleur. Les avantages procurés par 4D Compiler sont les suivants :

• **Vitesse :** votre base de données s'exécute de 3 à 1000 fois plus vite.

• **Vérification du code :** la cohérence interne du code de votre application de base de données est entièrement contrôlée. Les conflits de logique et de syntaxe sont détectés.

• **Protection :** une base compilée dispose des mêmes fonctionnalités que la base originale, à la différence près que la structure et les méthodes ne peuvent plus être visualisées ni modifiées délibérément ou par inadvertance. La compilation d'une base assure la protection de l'application.

• **Application indépendantes "double-cliquables" :** 4D Compiler permet de générer des applications indépendantes (sous Windows, des fichiers ".EXE") comportant leurs propres icônes.

Les commandes de ce thème sont liées à l'utilisation du compilateur. Elles vous permettent de normaliser les types de données exploitées dans votre base. La commande APPELER 4D est utilisée spécifiquement dans les bases compilées.

C_BLOB	C_ENTIER	C_REEL	APPELER 4D
C_BOOLEEN	C_ENTIER LONG	C_CHAINE	
C_DATE	C_IMAGE	C_TEXTE	
C_GRAPHE	C_POINTEUR	C_HEURE	

A l'exception d'APPELER 4D, ces commandes déclarent des variables et leur assignent un type. La déclaration des variables permet de lever toute ambiguité en ce qui concerne leur type. Lorsqu'une variable n'est pas déclarée par l'une de ces commandes, le compilateur déduit son type. Mais il lui est souvent difficile de déduire le type d'une variable utilisée dans un formulaire. Par conséquent, il est particulièrement important d'utiliser ces commandes pour déclarer les variables placées dans un formulaire.

Les opérations numériques effectuées sur des variables de type Entier long ou Entier sont généralement beaucoup plus rapides que celles effectuées sur des valeurs Numérique (réel).

Principes généraux d'écriture de code destiné à être compilé

• Les indirections de variables, utilisées dans la version 3 de 4D, ne sont pas permises. Vous ne pouvez pas utiliser l'indirection alphanumérique, à l'aide du symbole 'paragraphe' (§), pour référencer des variables indirectement. Vous ne pouvez pas non plus utiliser les indirections numériques, à l'aide des accolades ({...}). Les accolades ne peuvent être utilisées que pour accéder à un élément de tableau ayant été déclaré. En revanche, vous pouvez utiliser le passage de paramètres, comme décrit dans la documentation de 4D Compiler.

• Vous ne pouvez pas modifier le type d'une variable ou d'un tableau.

• Vous ne pouvez pas convertir un tableau simple en tableau à deux dimensions, et vice-versa.

• Vous ne pouvez pas modifier la longueur d'une variable chaîne ni celle des éléments d'un tableau alphanumérique.

• Bien que 4D Compiler déduise le type des variables si nécessaire, il est conseillé de déclarer le type des variables à l'aide des directives de compilation lorsque le type de données est ambigu, en particulier dans un formulaire.

• Une autre raison de déclarer explicitement le type des variables est l'optimisation de votre code. Cela est particulièrement vrai pour les variables utilisées comme compteurs. Dans ce cas, l'utilisation de variables de type Entier long assure un maximum de performances.

• Pour effacer une variable (c'est-à-dire l'initialiser à une valeur nulle), utilisez la commande EFFACER VARIABLE avec le nom de la variable. N'utilisez pas de chaîne alphanumérique pour désigner le nom de la variable avec la commande EFFACER VARIABLE.

• La fonction Indefinie retournera toujours Faux. Les variables sont toujours définies.

Exemples

(1) Voici quelques déclarations de variables standard pour 4D Compiler :

```
      ` La variable process vxMonBlob est déclarée avec le type BLOB
⇒     C_BLOB(vxMonBlob)
      ` La variable interprocess <>SousWindows est déclarée avec le type booléen
⇒     C_BOOLEEN(<>SousWindows)
      ` La variable locale $vdCurDate est déclarée avec le type Date
⇒     C_DATE($vdCurDate)
      ` Les trois variables process vg1, vg2 et vg3 sont déclarées avec le type Graphe
⇒     C_GRAPHE(vg1;vg2;vg3)
```

(2) Dans cet exemple, la méthode projet uneMéthodeParmiD'Autres déclare 3 paramètres:

```
` Méthode projet uneMéthodeParmiD'Autres
` uneMéthodeParmiD'Autres ( Numérique ; Entier { ; Entier long } )
` uneMéthodeParmiD'Autres ( Montant ; Pourcentage { ; Ratio } )
```

⇒ **C_REEL**($1) ` le 1er paramètre est du type Réel (Numérique)
⇒ **C_ENTIER**($2) ` le 2e paramètre est du type Entier
⇒ **C_ENTIER LONG**($3) ` le 3e paramètre est du type Entier long

` ...

(3) Dans l'exemple suivant, la méthode projet ajoutCapitale accepte un paramètre de type Chaîne et retourne une chaîne :

```
` Méthode projet ajoutCapitale
` ajoutCapitale ( Alpha ) -> Aplha
` ajoutCapitale ( Chaîne source ) -> Chaîne avec la première lettre capitale
```

⇒ **C_ALPHA**(255;$0;$1)
$0:=**Majusc**(**Sous chaine**($1;1;1))+**Minusc**(**Sous chaine**($1;2))

(4) Dans l'exemple suivant, la méthode projet envoyerPaquets accepte un paramètre de type Heure suivi d'un nombre variable de paramètres de type Texte :

```
` Méthode projet envoyerPaquets
` envoyerPaquets ( Heure ; Texte { ; Texte2... ; TextN } )
` envoyerPaquets ( docRef ; Données { ; Données2... ; DonnéesN } )
```

⇒ **C_HEURE** ($1)
⇒ **C_TEXTE** (${2})
⇒ **C_ENTIER LONG** ($vlPaquet)

```
Boucle ($vlPaquet;2;Nombre de parametres)
   ENVOYER PAQUET ($1;${$vlPaquet})
Fin de boucle
```

(5) Dans l'exemple suivant, la méthode projet compiler_Param_Prédéclare28 pré-déclare la syntaxe d'autres méthodes projet, à l'intention de 4D Compiler :

```
      ` Méthode projet compiler_Param_Prédéclare28

      ` uneMéthodeParmiD'Autres ( Réel ; Entier { ; Entier long } )
⇒     C_REEL(uneMéthodeParmiD'Autres;$1)
⇒     C_ENTIER(uneMéthodeParmiD'Autres;$2)        ` ...
⇒     C_ENTIER LONG(uneMéthodeParmiD'Autres;$3)        ` ...
⇒     C_ALPHA(ajoutCapitale;255;$0;$1)       ` ajoutCapitale ( Alpha ) -> Alpha
⇒     C_HEURE(envoyerPaquets;$1) ` envoyerPaquets ( Heure ; Texte {; Texte2... ; TexteN })
⇒     C_TEXTE(envoyerPaquets;${2})            ` ...
```

Référence

APPELER 4D, C_ALPHA, C_BLOB, C_BOOLEEN, C_DATE, C_ENTIER, C_ENTIER LONG, C_GRAPHE, C_HEURE, C_IMAGE, C_POINTEUR, C_REEL, C_TEXTE.

C_BLOB ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_BLOB assigne le type BLOB à toutes les variables spécifiées.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_BLOB(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_BLOB(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous aux exemples de la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur.

C_BOOLEEN ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_BOOLEEN affecte le type Booléen à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_BOOLEEN(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_BOOLEEN(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur, Nombre de parametres.

C_DATE ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_DATE affecte le type Date à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_DATE(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_DATE(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur, Nombre de parametres.

C_GRAPHE ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Alpha	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_GRAPHE affecte le type Graphe à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_GRAPHE(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_GRAPHE(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Référez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur.

C_ENTIER ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_ENTIER affecte le type Entier à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_ENTIER(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_ENTIER(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur, C_ENTIER LONG, C_REEL, Nombre de parametres.

C_ENTIER LONG

Compilateur

version 5

C_ENTIER LONG ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_ENTIER LONG affecte le type Entier long à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_ENTIER LONG(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_ENTIER LONG(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur, C_ENTIER, C_REEL, Nombre de parametres.

C_IMAGE ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_IMAGE affecte le type Image à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_IMAGE(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_IMAGE(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur, Nombre de parametres.

C_POINTEUR

Compilateur

version 5

C_POINTEUR ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_POINTEUR affecte le type Pointeur à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_POINTEUR(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_POINTEUR(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur, Nombre de parametres.

C_REEL ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_REEL affecte le type Réel (numérique) à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_REEL(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_REEL(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur, C_ENTIER, C_ENTIER LONG, Nombre de parametres.

C_ALPHA ({méthode; }taille; variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
taille	Numérique	→	Taille de la chaîne
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_ALPHA affecte le type Alphanumérique à chaque variable spécifiée.

Le paramètre taille spécifie la longueur maximum des chaînes qu'une variable peut contenir. Les chaînes sont limitées à 255 caractères. Si vous souhaitez optimiser la vitesse d'exécution de votre base, il est préférable d'utiliser des variables Alpha au lieu de variables Texte lorsque c'est possible.

La première syntaxe de la commande (lorsque le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (lorsque le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe pour éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_ALPHA(...;${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_ALPHA(...;${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

C_TEXTE ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_TEXTE affecte le type Texte à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_TEXTE(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_TEXTE(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur, C_ALPHA, Nombre de parametres.

C_HEURE ({méthode; }variable{; variable2; ...; variableN})

Paramètre	Type		Description
méthode	Méthode	→	Nom de méthode
variable	Variable ou ${...}	→	Nom(s) de(s) variable(s) à déclarer

Description

C_HEURE affecte le type Heure à chaque variable spécifiée.

La première syntaxe de la commande (si le paramètre méthode n'est pas passé) est utilisée pour déclarer et typer une variable process, interprocess ou locale.

Note : Cette syntaxe peut être utilisée dans les bases interprétées.

La seconde syntaxe de la commande (si le paramètre méthode est passé) est utilisée pour déclarer d'avance pour 4D Compiler le résultat et/ou les paramètres ($0, $1, $2, etc.) d'une méthode. Vous devez utiliser cette syntaxe si vous voulez éviter la phase de typage de variables lors de la compilation d'une base, afin de réduire le temps de compilation.

ATTENTION : La deuxième syntaxe ne peut pas être exécutée en mode interprété. Pour cette raison, si vous utilisez cette syntaxe, stockez-la dans une méthode (dont le nom doit débuter par "COMPILER") qui n'est pas exécutée en mode interprété.

Utilisation avancée : La syntaxe C_HEURE(${...}) vous permet de déclarer pour une méthode un nombre variable de paramètres du même type à la condition que ce soient les derniers paramètres de la méthode. Par exemple, la déclaration C_HEURE(${5}) indique à 4D et à 4D Compiler qu'à partir du cinquième paramètre la méthode peut recevoir un nombre variable de paramètres de ce type. Pour plus d'informations, référez-vous à la commande Nombre de parametres.

Exemples

Reportez-vous à la section Commandes du thème Compilateur.

Référence

Commandes du thème Compilateur, Nombre de parametres.

APPELER 4D

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

APPELER 4D est destinée uniquement à une utilisation avec 4D Compiler. En effet, seul le moteur de 4e Dimension peut détecter un événement. Il était donc nécessaire, dans le cadre d'une base compilée, qu'une routine puisse interroger le moteur de 4e Dimension afin de savoir si un événement s'est produit. Cette commande doit donc être utilisée lorsque vous employez la commande APPELER SUR EVENEMENT.
Par exemple, si une méthode exécute une boucle dans laquelle aucune commande 4e Dimension n'est appelée, la boucle ne pourra pas être interrompue par un process installé à l'aide d'APPELER SUR EVENEMENT, et l'utilisateur ne pourra pas ouvrir une autre application. Dans ce cas, APPELER 4D doit être insérée pour que 4e Dimension puisse intercepter les événements. Bien entendu, n'utilisez pas APPELER 4D si vous ne voulez aucune interruption.

Exemple

Dans l'exemple suivant, la boucle ne se terminerait jamais dans une base compilée sans l'aide de APPELER 4D :

```
    ` Méthode Traitement quelconque
APPELER SUR EVENEMENT ("METHODE EVENEMENT")
◊vbArrêt:=Faux
MESSAGE ("Traitement..."+Caractere(13)+"Tapez une touche pour
                                        interrompre l'exécution...")
Repeter
    ` Effectuer un traitement sans appel à une commande 4D
⇒   APPELER 4D
Jusque (◊vbArrêt)
APPELER SUR EVENEMENT ("")
```

La méthode METHODE EVENEMENT :

```
   ` Méthode METHODE EVENEMENT
Si (Indefinie(CodeTouche))
   CodeTouche:=0
Fin de si
Si (CodeTouche#0)
   CONFIRMER ("Voulez-vous vraiment interrompre cette opération ?")
   Si (OK=1)
      ◊vbArrêt:=Vrai
   Fin de si
Fin de si
```

Référence

APPELER SUR EVENEMENT, Commandes du thème Compilateur.

9

Dates et heures

Date du jour {(*)} → Date

Paramètre	Type		Description
*		→	Retourne la date du jour du serveur
Résultat	Date	←	Date du jour

Description

Date du jour retourne la date courante telle que définie dans l'horloge système de la machine.

4D Server : Si vous passez le paramètre astérisque (*) — lors d'une exécution sur un poste 4D Client —, la fonction retourne la date du jour telle que définie dans l'horloge du poste serveur.

Exemples

(1) L'exemple suivant fait apparaître une boîte de dialogue d'alerte affichant la date du jour :

```
⇒ ALERTE("Nous sommes le " + Chaine(Date du jour) + ".")
```

(2) Vous développez une application pour le marché international. Vous souhaitez savoir si la version de 4D avec laquelle votre application est exécutée fonctionne avec des dates formatées en MM/JJ/AAAA (version US) ou JJ/MM/AAAA (version française). Cette information est nécessaire pour vous permettre, par exemple, de personnaliser correctement les zones de saisie.

La méthode projet suivante vous permet de traiter cette question :

```
` Méthode projet (fonction) Format date système
` Format date système -> Chaine
` Format date système -> Format de données 4D par défaut

C_ALPHA(31;$0;$vsDate;$vsMJA;$vsMois;$vsJour;$vsAnnée)
C_ENTIER LONG($1;$vlPos)
C_DATE($vdDate)
```

```
   ` Récupérer une date dans laquelle les valeurs de mois, de jour et d'année sont toutes
   ` différentes
⇒  $vdDate:=Date du jour
   Repeter
      $vsMois:=Chaine(Mois de($vdDate))
      $vsJour:=Chaine(Jour de($vdDate))
      $vsAnnée:=Chaine(Annee de($vdDate)%100)
      Si (($vsMois=$vsJour) | ($vsMois=$vsAnnée) | ($vsJour=$vsAnnée))
         OK:=0
         $vdDate:=$vdDate+1
      Sinon
         OK:=1
      Fin de si
   Jusque (OK=1)
   $0:="" ` Initialisation du résultat de la fonction
   $vsDate:=Chaine($vdDate)
   $vlPos:=Position("/";$vsDate) ` Trouver le premier séparateur / dans la chaîne ../../..
   $vsMJA:=Sous chaine($vsDate;1;$vlPos-1) ` Extraire les premiers chiffres de la date
      ` Eliminer les premiers chiffres et le premier séparateur /
   $vsDate:=Sous chaine($vsDate;$vlPos+1)
   Au cas ou
      : ($vsMJA=$vsMois) ` Les chiffres expriment le mois
         $0:="MM"
      : ($vsMJA=$vsJour) ` Les chiffres expriment le jour
         $0:="JJ"
      : ($vsMJA=$vsAnnée) ` Les chiffres expriment l'année
         $0:="AAAA"
   Fin de cas
   $0:=$0+"/" ` Commencer à construire le résultat de la fonction
   $vlPos:=Position("/";$vsDate) ` Trouver le deuxième séparateur dans la chaîne ../..
   $vsMJA:=Sous chaine($vsDate;1;$vlPos-1) ` Extraire les chiffres suivants de la date
      ` Réduire la chaîne aux derniers chiffres de la date
   $vsDate:=Sous chaine($vsDate;$vlPos+1)
   Au cas ou
      : ($vsMJA=$vsMois) ` Les chiffres expriment le mois
         $0:=$0+"MM"
      : ($vsMJA=$vsJour) ` Les chiffres expriment le jour
         $0:=$0+"JJ"
      : ($vsMJA=$vsAnnée) ` Les chiffres expriment l'année
         $0:=$0+"AAAA"
   Fin de cas
   $0:=$0+"/" ` Poursuivre la construction du résultat de la fonction
```

```
Au cas ou
   : ($vsDate=$vsMois) ` Les chiffres expriment le mois
      $0:=$0+"MM"
   : ($vsDate=$vsJour) ` Les chiffres expriment le jour
      $0:=$0+"DD"
   : ($vsDate=$vsAnnée) ` Les chiffres expriment l'année
      $0:=$0+"AAAA"
Fin de cas
   ` A ce moment, $0 vaut soit MM/JJ/AAAA soit JJ/MM/AAAA, ou encore...
```

Référence

Annee de, Jour de, Mois de, Opérateurs sur les dates.

Jour de (date) → Numérique

Paramètre	Type		Description
date	Date	→	Date dont vous voulez extraire le jour
Résultat	Numérique	←	Jour du mois de date

Description

Jour de retourne le jour du mois de date.

Exemples

(1) L'exemple suivant illustre l'utilisation de Jour de. Les valeurs retournées sont stockées dans la variable Résultat. Les commentaires décrivent la valeur de Résultat :

```
⇒ Résultat := Jour de (!25/12/96!) ` Résultat vaut 25
⇒ Résultat := Jour de (Date du jour) ` Résultat prend la valeur du jour d'aujourd'hui
```

(2) Reportez-vous à l'exemple de la fonction Date du jour.

Référence

Annee de, Mois de, Numero du jour.

Mois de (date) → Numérique

Paramètre	Type		Description
date	Date	→	Date dont vous voulez extraire le mois
Résultat	Numérique	←	Nombre indiquant le mois de date

Description

Mois de retourne un nombre représentant le numéro du mois de date.

Note : C'est le numéro du mois est retourné, et non son nom (reportez-vous à l'exemple ci-dessous).

4e Dimension fournit les constantes prédéfinies suivantes :

Constantes	Type	Valeur
Janvier	Entier long	1
Février	Entier long	2
Mars	Entier long	3
Avril	Entier long	4
Mai	Entier long	5
Juin	Entier long	6
Juillet	Entier long	7
Août	Entier long	8
Septembre	Entier long	9
Octobre	Entier long	10
Novembre	Entier long	11
Décembre	Entier long	12

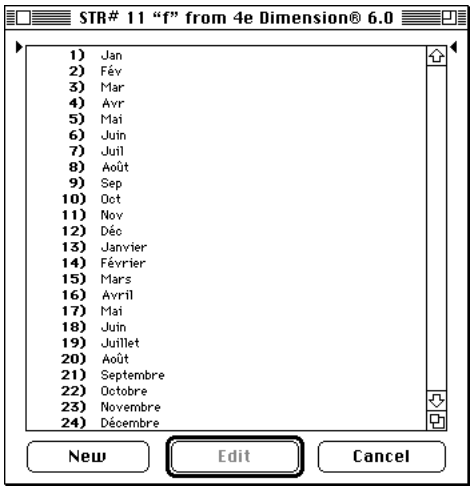
Exemples

(1) L'exemple suivant illustre l'utilisation de Mois de. Les valeurs retournées sont assignées à la variable Résultat. Les commentaires fournissent les valeurs de Résultat :

```
⇒ Résultat := Mois de (!25/12/96!) ` Résultat vaut 12
⇒ Résultat := Mois de (Date du jour) ` Résultat prend la valeur du mois d'aujourd'hui
```

(2) Reportez-vous à l'exemple de la fonction Date du jour.

(3) La ressource 'STR#' n°11 de 4e Dimension contient les noms des mois dans la langue courante de l'application :



La méthode projet suivante retourne le nom du mois pour une date :

```
` Méthode projet Nom du mois
` Nom du mois ( Date ) -> Chaîne
` Nom du mois ( Date ) -> Nom du mois

$0:=Lire chaine dans liste(11;12+Mois de ($1))
```

⇒ $0:=**Lire chaine dans liste**(11;12+**Mois de** ($1))

La méthode projet suivante retourne le nom du mois sous forme abrégée pour une date :

```
` Méthode projet Mois abrégé
` Mois abrégé ( Date ) -> Chaîne
` Mois abrégé ( Date ) -> Nom abrégé du mois

$0:=Lire chaine dans liste(11;Mois de ($1))
```

⇒ $0:=**Lire chaine dans liste**(11;**Mois de** ($1))

Référence

Annee de, Jour de.

Annee de (date) → Numérique

Paramètre	Type		Description
date	Date	→	Date dont vous voulez extraire l'année
Résultat	Numérique	←	Nombre indiquant l'année de date

Description

Annee de retourne un nombre indiquant l'année de date.

Exemples

(1) L'exemple suivant illustre l'utilisation de Annee de. Les résultats sont assignés à la variable Résultat :

```
⇒ Résultat := Annee de (!25/12/96!)  ` Résultat prend la valeur 1996
⇒ Résultat := Annee de (!25/12/1996!)  ` Résultat prend la valeur 1996
⇒ Résultat := Annee de (!25/12/1896!)  ` Résultat prend la valeur 1896
⇒ Résultat := Annee de (!25/12/2096!)  ` Résultat prend la valeur 2096
     ` Résultat prend comme valeur l'année de la date d'aujourd'hui
⇒ Résultat := Annee de (Date du jour)
```

(2) Reportez-vous à l'exemple de la fonction Date du jour.

Référence

Jour de, Mois de.

Numero du jour (date) → Numérique

Paramètre	Type		Description
date	Date	→	Date dont vous souhaitez connaître le numéro du jour
Résultat	Numérique	←	Numéro représentant le jour de la semaine auquel date correspond

Description

La fonction Numero du jour retourne un numéro représentant le jour de la semaine auquel date correspond.

Note : Si une date nulle est passée à Numero du jour, la fonction retourne 2.

4e Dimension fournit les constantes prédéfines suivantes :

Constantes	Type	Valeur
Lundi	Entier long	2
Mardi	Entier long	3
Mercredi	Entier long	4
Jeudi	Entier long	5
Vendredi	Entier long	6
Samedi	Entier long	7
Dimanche	Entier long	1

Note : Numero du jour retourne une valeur comprise entre 1 et 7. Pour obtenir le numéro du jour dans le sens "date du mois", utilisez la fonction Jour de.

Exemple

L'exemple suivant est une fonction qui retourne le jour d'aujourd'hui sous forme de chaîne :

```
   ` $Jour prend comme valeur le numéro du jour courant
⇒  $Jour := Numero du jour (Date du jour)
   Au cas ou
      : ($Jour = 1)
      $0 := "Dimanche"
      : ($Jour = 2)
      $0 := "Lundi"
      : ($Jour = 3)
      $0 := "Mardi"
      : ($Jour = 4)
      $0 := "Mercredi"
      : ($Jour = 5)
      $0 := "Jeudi"
      : ($Jour = 6)
      $0 := "Vendredi"
      : ($Jour = 7)
      $0 := "Samedi"
   Fin de cas
```

Référence

Jour de.

Ajouter a date (date; années; mois; jours) → Date

Paramètre	Type		Description
date	Date	→	Date à laquelle ajouter jours, mois et années
années	Numérique	→	Nombre d'années à ajouter à la date
mois	Numérique	→	Nombre de mois à ajouter à la date
jours	Numérique	→	Nombre de jours à ajouter à la date
Résultat	Date	←	Date résultante

Description

Ajouter a date ajoute années, mois et jours à la date que vous avez passée dans date, et retourne la date résultante.

Alors que les Opérateurs sur les dates vous permettent d'ajouter des jours à une date, Ajouter a date vous permet d'ajouter rapidement des mois et des années sans vous soucier du nombre de jours par mois ou des années bissextiles (comme vous devriez le faire avec l'opérateur "+" sur les dates).

Exemples

```
   ` Cette ligne calcule la date dans un an, le même jour
⇒  $vdDansUnAn:=Ajouter a date(Date du jour;1;0;0)

   ` Cette ligne calcule la date le mois prochain, le même jour
⇒  $vdMoisProchain:=Ajouter a date(Date du jour;0;1;0)

   ` Cette ligne fait la même chose que $vdDemain:=Date du jour+1
⇒  $vdDemain:=Ajouter a date(Date du jour;0;0;1)
```

Référence

Opérateurs sur les dates.

Date (chaîneDate) → Date

Paramètre	Type		Description
chaîneDate	Alpha	→	Chaîne contenant la date à retourner
Résultat	Date	←	chaîneDate sous forme de Date

Description

La fonction Date extrait et retourne la date de la chaîne chaîneDate.

Le paramètre chaîneDate doit respecter les conventions d'écriture standard pour les formats de date. Dans une version française de 4D, la date doit être de la forme JJ/MM/AA (jour, mois, année). Le jour et le mois peuvent être composés d'un ou deux chiffres. L'année peut être composée de deux ou quatre chiffres. Si l'année comporte deux chiffres, Date considère que la date appartient au XXe siècle et ajoute 19 devant la valeur. Les caractères de séparation de date autorisés sont les suivants : barre oblique (/), espace, point (.), virgule (,) et tiret (-).

Date ne vérifie pas la validité de la date passée dans chaîneDate. Si une date erronée (telle que "13/35/94") est passée, Date retourne une date invalide. Si chaîneDate ne peut être interprétée comme une date (par exemple, "aa/12/94"), une date nulle (!00/00/00!) est retournée.

Il est de votre ressort de tester la validité de chaîneDate.

Exemples

(1) L'exemple suivant demande à l'utilisateur de saisir une date. La chaîne saisie est convertie en date et stockée dans la variable DemDate :

```
⇒   DemDate := Date (Demander ("Saisissez une date :"; Chaine (Date du jour)))
    Si (OK=1)
        ` Faire quelque chose avec la date
    Fin de si
```

(2) L'exemple suivant retourne la chaîne "12/12/97" sous forme de date :

```
⇒   vDate:=Date("12/12/97")
```

Heure courante {(*)} → Heure

Paramètre	Type		Description
*		→	Retourne l'heure courante sur le poste serveur
Résultat	Heure	←	Heure courante

Description

La fonction Heure courante retourne l'heure courante définie dans l'horloge de votre système.

L'heure courante est toujours comprise entre 00:00:00 et 23:59:59. Vous pouvez utiliser les fonctions Chaine ou Chaine heure pour convertir en chaîne alphanumérique l'expression de type heure retournée par Heure courante.

4D Server : Si vous passez le paramètre astérisque (*) — lors d'une exécution sur un poste 4D Client —, la fonction retourne l'heure courante telle que définie dans l'horloge du poste serveur.

Exemples

(1) L'exemple suivant vous permet de mesurer la durée d'une opération. Dans cet exemple, vous voulez chronométrer la méthode longueOpération :

```
⇒   CelaPrend := Heure courante  ` Stocker l'heure de départ
    longueOpération  ` Effectuer l'opération
        ` Afficher la durée
    ALERTE ("L'opération a pris " + Chaine (Heure courante – CelaPrend))
```

(2) L'exemple suivant extrait les heures, minutes et secondes de l'heure courante :

```
⇒   $vhMaintenant:= Heure courante
    ALERTE("L'heure courante est : "+Chaine($vhMaintenant\3600))
    ALERTE("La minute courante est : "+Chaine(($vhMaintenant\60)%60))
    ALERTE("La seconde courante est : "+Chaine($vhMaintenant%60))
```

Référence

Chaine, Nombre de millisecondes, Nombre de ticks, Opérateurs sur les heures.

Chaine heure (secondes) → Alpha

Paramètre	Type		Description
secondes	Numérique	→	Secondes écoulées depuis minuit
Résultat	Alpha	←	Heure sous forme de chaîne au format 24 heures

Description

La fonction Chaine heure retourne sous forme de chaîne alphanumérique sur 24 heures l'expression de type Heure passée dans secondes.

Le format appliqué à la chaîne est HH:MM:SS.

Si vous passez un nombre de secondes supérieur à celui qu'il y a dans un jour (86 400), Chaine heure continue d'ajouter les heures, les minutes et les secondes. Par exemple, Chaine heure (86401) retourne 24:00:01.

Note : Si vous voulez obtenir sous forme de chaîne une expression de type Heure dans des formats plus variés, utilisez la fonction Chaine.

Exemple

L'exemple suivant affiche une boîte de dialogue d'alerte avec le message “46800 secondes représentent 13:00:00” :

```
⇒ ALERTE ("46800 secondes représentent " + Chaine heure (46800))
```

Référence

Chaine, Heure.

Heure (chaineHeure) → Heure

Paramètre	Type		Description
chaineHeure	Alpha	→	Chaîne à retourner sous forme d'heure
Résultat	Heure	←	Heure définie par chaineHeure

Description

La fonction Heure retourne, sous la forme d'une expression de type Heure, l'heure définie dans la chaîne chaineHeure.

Le paramètre chaineHeure doit être de la forme HH:MM:SS et dans un format de 24 heures.

Exemple

L'exemple suivant affiche une boîte de dialogue d'alerte avec le message “1:00 P.M. = 13 heures 0 minute.” :

```
⇒ ALERTE ("1:00 P.M. = " + Chaine (Heure ("13:00:00"); Heure Minute))
```

Référence

Chaine, Chaine heure.

Nombre de ticks → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Nombre de ticks (60ème de seconde) écoulés depuis le démarrage de la machine

Description

Nombre de ticks retourne le nombre de ticks (1 tick = 1/60ème de seconde) écoulés depuis le démarrage de la machine.

Note : Nombre de ticks retourne une valeur de type Entier long.

Exemple

Référez-vous à l'exemple de la fonction Nombre de millisecondes.

Référence

Heure courante, Nombre de millisecondes.

Nombre de millisecondes → Entier long

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	Nombre de millisecondes (1000ème de seconde) écoulées depuis le démarrage de la machine

Description

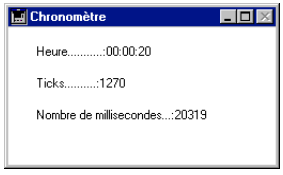
Nombre de millisecondes retourne le nombre de millisecondes (1 milliseconde = 1/1000ème de seconde) écoulées depuis le démarrage de la machine.

Exemple

Le code suivant :

```
      Creer fenetre (100;100;350;230;0;"Chronomètre")
      $vhDébutHeure:=Heure courante
      $vlDébutTicks:=Nombre de ticks
⇒     $vrDébutMillisecondes:=Nombre de millisecondes
      Repeter
         POSITION MESSAGE (2;1)
         MESSAGE ("Heure.....:"+Chaine (Heure courante -$vhDébutHeure))
         POSITION MESSAGE (2;3)
         MESSAGE ("Ticks.....:"+Chaine (Nombre de ticks -$vlDébutTicks))
         POSITION MESSAGE (2;5)
⇒        MESSAGE ("Nombre de millisecondes...:"+Chaine (Nombre de
                                                millisecondes -$vrDébutMillisecondes))
      Jusque ((Heure courante -$vhDébutHeure)>=†00:01:00†)
      FERMER FENETRE
```

... affiche la fenêtre suivante pendant une minute :



Référence

Heure courante, Nombre de ticks.

SIECLE PAR DEFAUT

Dates et heures

version 6.7 (Modifiée)

SIECLE PAR DEFAUT (siècle{; anPivot})

Paramètre	Type		Description
siècle	Numérique	→	Siècle par défaut (moins un) lors de la saisie d'années sur 2 chiffres
anPivot	Numérique	→	Année pivot lors de la saisie d'années sur 2 chiffres

Description

La commande SIECLE PAR DEFAUT vous permet de définir le siècle courant par défaut et l'année pivot adoptés par 4D lorsque des dates sont saisies avec seulement deux chiffres pour l'année.

L'année pivot indique la valeur au-dessous de laquelle une année saisie sur deux chiffres sera interprétée comme appartenant au siècle suivant :
• si l'année saisie est supérieure ou égale à l'année pivot, elle appartient au siècle courant,
• si l'année saisie est inférieure à l'année pivot, elle appartient au siècle suivant (relativement au siècle courant par défaut).

Par défaut, 4D présume que les dates appartiennent au 20e siècle et utilise la valeur 30 comme année pivot :
• Si vous saisissez la date 25/01/97, 4D considère que vous indiquez le 25 janvier 1997
• Si vous saisissez la date 25/01/30, 4D considère que vous indiquez le 25 janvier 1930
• Si vous saisissez la date 25/01/29, 4D considère que vous indiquez le 25 janvier 2029
• Si vous saisissez la date 25/01/07, 4D considère que vous indiquez le 25 janvier 2007

La commande SIECLE PAR DEFAUT permet de modifier ce comportement par défaut. Une fois exécutée, elle prend effet immédiatement.
Vous pouvez passer uniquement un siècle par défaut, ou un siècle par défaut et une année pivot.

Si vous passez uniquement un nouveau siècle par défaut (moins un) dans siècle, 4D interprétera toutes les années saisies sur deux chiffres comme appartenant à ce siècle. Par exemple, après l'appel :

```
SIECLE PAR DEFAUT(20) ` Fixer le 21e siècle comme siècle par défaut
```

• Si vous saisissez la date 25/01/97, 4D considère que vous indiquez le 25 janvier 2097
• Si vous saisissez la date 25/01/07, 4D considère que vous indiquez le 25 janvier 2007

En outre, vous pouvez spécifier le paramètre anPivot.
Par exemple, après l'appel :

```
` Fixer le 21e siècle comme siècle par défaut si l'année est inférieure à 95
SIECLE PAR DEFAUT(19;95)
```

• Si vous saisissez la date 25/01/97, 4D considère que vous indiquez le 25 janvier 1997
• Si vous saisissez la date 25/01/07, 4D considère que vous indiquez le 25 janvier 2007

Notez que cette commande n'affecte que l'interprétation des dates lorsque les années sont saisies sur 2 chiffres. Quels que soient les paramètres passés à SIECLE PAR DEFAUT :
• Si vous saisissez la date 25/01/1997, 4D considère que vous indiquez le 25 janvier 1997
• Si vous saisissez la date 25/01/2097, 4D considère que vous indiquez le 25 janvier 2097
• Si vous saisissez la date 25/01/1907, 4D considère que vous indiquez le 25 janvier 1907
• Si vous saisissez la date 25/01/2007, 4D considère que vous indiquez le 25 janvier 2007

Cette commande affecte uniquement la saisie des données. Elle n'influe pas sur le stockage des données, les calculs, etc.

10

Débogueur

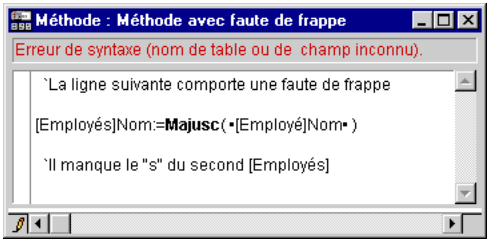
Un débogueur, pour quoi faire ?

Lorsque vous développez et testez vos méthodes, il est important que vous puissiez repérer et corriger les erreurs qui peuvent s'y être glissées.

Vous pouvez commettre plusieurs types d'erreurs en utilisant le langage : des fautes de frappe, des erreurs de syntaxe, des erreurs liées à l'environnement, des erreurs logiques ou de conception, ou encore des erreurs d'exécution (Runtime).

Fautes de frappe

Les fautes de frappe sont détectées directement dans l'**éditeur de méthodes** et sont indiquées par une puce (•) ainsi qu'un message dans la zone d'information située en haut de la fenêtre. Cette fenêtre signale une faute de frappe :



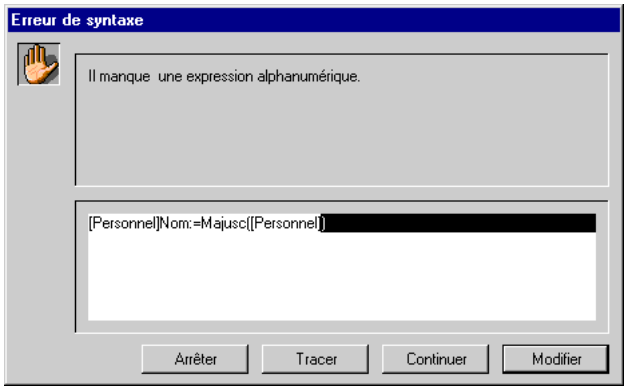
Note : Les commentaires ont été manuellement ajoutés pour cet exemple. Seules les puces (•) sont insérées par 4D à l'endroit de l'erreur.

De telles fautes de frappe provoquent généralement des erreurs de syntaxe (dans ce cas, le nom de la table est inconnu). La zone d'informations de la fenêtre affiche la cause de l'erreur au moment de la validation de la ligne d'instructions.

Vous pouvez alors corriger l'erreur de frappe et appuyer sur la touche **Entrée** (du clavier numérique) pour valider la correction. Pour plus d'informations sur l'éditeur de méthodes, reportez-vous au manuel *Mode Structure* de 4e Dimension.

Erreurs de syntaxe

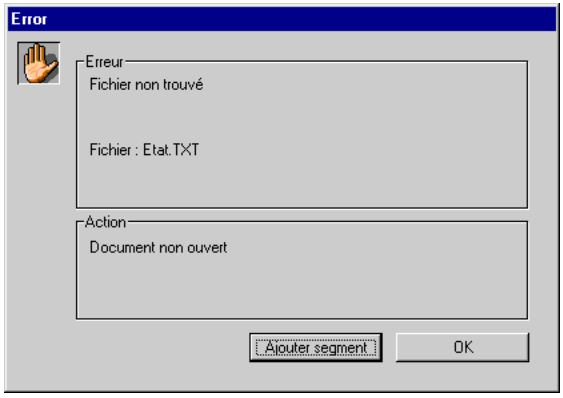
Certaines erreurs de syntaxe ne peuvent être détectées que lorsque vous exécutez la méthode. La fenêtre Erreur de syntaxe est affichée lorsqu'une telle erreur de syntaxe est détectée. Par exemple :



Ici, l'erreur provient du fait que le nom d'une table est passé à la commande Majusc, qui attend une expression de type Texte. Pour plus d'informations sur cette fenêtre, reportez-vous à la section Fenêtre d'erreur de syntaxe.

Erreurs liées à l'environnement

Parfois, il peut arriver que vous n'ayez pas assez de mémoire pour créer un tableau ou un BLOB. Ou bien, lorsque vous cherchez à accéder à un document sur votre disque, ce document peut ne pas exister ou être déjà ouvert par une autre application. Dans ce cas, une fenêtre d'erreur apparaît, vous décrivant l'erreur et l'action qui n'a pas pu être effectuée. Par exemple :



Ces erreurs ne se produisent pas à cause de votre code. Simplement, ce sont des choses qui arrivent ! La plupart d'entre elles sont faciles à identifier à l'aide d'une méthode d'interception d'erreurs. Pour plus d'informations, reportez-vous à la description de la commande APPELER SUR ERREUR.

Erreurs logiques ou de conception

Ce sont généralement les erreurs qui sont les plus difficiles à repérer. Vous devez utiliser le débogueur pour les détecter. Notez qu'à l'exception des fautes de frappe, tous les types d'erreurs indiqués précédemment sont aussi, parfois, des erreurs de logique ou de conception. Voici des exemples :

• Une erreur de syntaxe peut être retournée si vous essayez d'utiliser une variable qui n'est pas encore initialisée.
• Une erreur liée à l'environnement peut se produire si vous essayez d'ouvrir un document dont le nom est reçu par une sous-routine qui ne reçoit pas la bonne valeur dans le paramètre. Notez que dans cet exemple, le morceau de code qui ne fonctionne pas n'est pas celui qui est à l'origine du problème.

Les erreurs logiques ou de conception se produisent également dans les situations suivantes :

• Un enregistrement n'est pas correctement mis à jour parce qu'en appelant STOCKER ENREGISTREMENT, vous avez oublié de tester d'abord si cet enregistrement était ou non verrouillé.
• Une méthode ne fait pas exactement ce à quoi vous vous attendez parce que vous ne testez pas la présence éventuelle d'un paramètre optionnel.

Erreurs d'exécution

En mode compilé, vous pouvez obtenir des erreurs que vous n'avez jamais vues en mode interprété. Voici un exemple :

Ce message vous indique que vous essayez d'accéder à un caractère dont la position se trouve au-delà de la longueur de la chaîne. Pour trouver rapidement l'origine du problème, notez le nom de la méthode et le numéro de la ligne, ouvrez votre base en mode interprété puis allez à la méthode et à la ligne indiquées.

Que faire lorsque vous rencontrez une erreur ?

Les erreurs sont chose commune. Il est rare d'écrire une grande quantité de lignes de code sans générer d'erreur. Traiter et corriger les erreurs est tout aussi naturel.

Grâce à son environnement multitâche, 4D vous permet de modifier et d'exécuter rapidement vos méthodes : vous n'avez qu'à passer d'une fenêtre à une autre. De plus, vous découvrirez combien il est simple et rapide de repérer vos erreurs en utilisant le Débogueur.

Un réflexe courant chez les débutants lorsqu'une erreur est détectée est de cliquer sur le bouton **Arrêter** dans la fenêtre d'erreur de syntaxe, de retourner à l'éditeur de méthodes, et d'essayer de deviner ce qui se passe en regardant le code. Ne faites pas cela ! Vous gagnerez un temps considérable en utilisant **toujours** le Débogueur.

- S'il se produit une erreur de syntaxe inattendue, utilisez le Débogueur.
- S'il se produit une erreur d'environnement, utilisez le Débogueur.
- S'il se produit tout autre type d'erreur, utilisez le Débogueur.

Dans 99 % des cas, le Débogueur affiche l'information dont vous avez besoin pour comprendre la cause de l'erreur. Vous pouvez alors la corriger.

Astuce : Passez quelques heures à apprendre et expérimenter le Débogueur. Vous gagnerez des journées et des mois dans l'avenir.

La phase de développement représente une autre occasion d'utiliser le Débogueur. Imaginons que vous deviez écrire un algorithme plus complexe que d'habitude. Une fois le code écrit, vous ne pouvez être certain qu'il fonctionne tant que vous ne l'aurez pas testé. Au lieu de passer directement en exécution, vous pouvez d'abord le vérifier en insérant la commmande TRACE au début de votre code. Ensuite, exécutez le code au pas à pas pour contrôler ce qui se passe.

Conclusion générale

Utilisez le Débogueur.

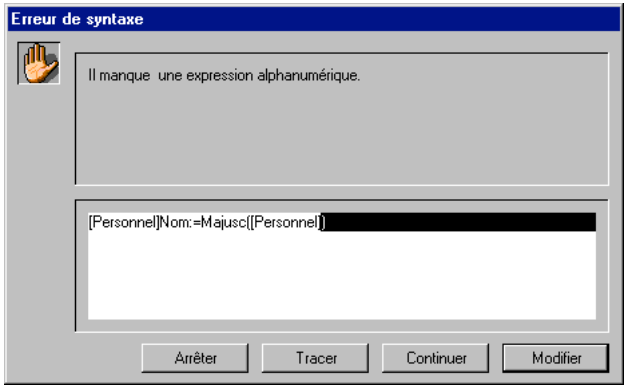
Référence

APPELER SUR ERREUR, Débogueur, Fenêtre d'erreur de syntaxe, Liste des points d'arrêt, Points d'arrêt sur commandes, Raccourcis du débogueur.

La Fenêtre d'erreur de syntaxe s'affiche lorsque l'exécution d'une méthode est interrompue. L'exécution de la méthode peut être interrompue pour l'une des deux raisons suivantes :

• 4D interrompt la méthode car une erreur de syntaxe l'empêche de poursuivre son exécution.
• Vous provoquez une interruption utilisateur en appuyant sur **Alt**+**Clic** (sous Windows) ou **Option**+**Clic** (sous MacOS) pendant l'exécution de la méthode.

Voici une Fenêtre d'erreur de syntaxe :



Le texte situé dans la zone supérieure de la fenêtre affiche un message décrivant l'erreur. La partie inférieure fait apparaître la ligne exécutée au moment où l'erreur est survenue ; l'emplacement précis où est survenue l'erreur est sélectionné.

La fenêtre comporte quatre boutons : **Arrêter**, **Tracer**, **Continuer** et **Modifier**.

• **Arrêter** : La méthode est interrompue et vous retournez à l'endroit où vous vous trouviez avant de commencer l'exécution de la méthode. Si une méthode formulaire ou une méthode objet s'exécutent en réponse à un événement, elles sont stoppées et vous retournez au formulaire. Si la méthode s'exécute à partir du mode Menus créés, vous retournez dans ce mode.

• **Tracer** : Vous entrez dans le mode Trace et la fenêtre du Débogueur est affichée. Si la ligne courante a été partiellement exécutée, il se peut que vous soyez obligé de cliquer plusieurs fois sur le bouton **Tracer**. Lorsque la ligne est terminée, la fenêtre du Débogueur s'affiche.

• **Continuer** : L'exécution continue. La ligne contenant l'erreur peut avoir été partiellement exécutée — tout dépend de l'endroit où se trouvait l'erreur. Continuez avec prudence — l'erreur peut empêcher que le reste de la méthode s'exécute correctement. Généralement, il vaut mieux ne pas continuer. Vous pouvez cliquer sur Continuer si l'erreur se trouve dans un appel mineur, comme par exemple CHANGER TITRE FENETRE, qui n'empêche pas de continuer l'exécution et le test du code. Vous pouvez vous concentrer sur le code le plus important, et corriger les erreurs mineures ultérieurement.

• **Modifier** : L'exécution de la méthode est totalement interrompue. 4D passe en mode Structure. La méthode dans laquelle l'erreur est survenue est ouverte dans l'éditeur de méthodes, ce qui vous permet de la corriger. Utilisez cette option lorsque vous avez identifié immédiatement l'erreur et que vous pouvez la corriger sans qu'il soit nécessaire d'effectuer d'autres investigations.

Référence

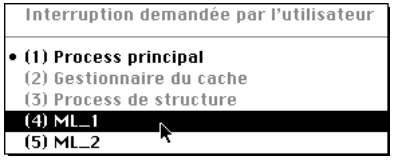
APPELER SUR ERREUR, Débogueur, Un débogueur, pour quoi faire ?.

version 6.5 (Modifiée)

Le mot Débogueur provient du terme anglais *bug* (insecte ou punaise) qui se "traduit" en français par *bogue* . Une bogue dans une méthode est une erreur que vous désirez éliminer. Lorsqu'une erreur se produit dans votre code, ou lorsque vous désirez contrôler l'exécution de vos méthodes, vous utilisez le débogueur. Un débogueur vous permet de trouver les bogues en exécutant pas à pas vos méthodes et en fournissant toutes les informations nécessaires. Ce procédé s'appelle le **mode Trace**.

Pour appeler la fenêtre du débogueur et afficher puis tracer les méthodes, vous pouvez :

• Cliquer sur le bouton **Tracer** dans la fenêtre d'erreur de syntaxe,
• Utiliser la commande TRACE,
• Cliquer sur le bouton **Déboguer**, en mode Utilisation, dans la fenêtre d'exécution de méthode.
• Utiliser la combinaison **Alt**+**Maj**+**clic droit** (sous Windows) ou **Control**+**Option**+**Commande**+**clic** (sous MacOS) pendant l'exécution d'une méthode, puis choisir le process à tracer dans le pop up menu qui apparaît :

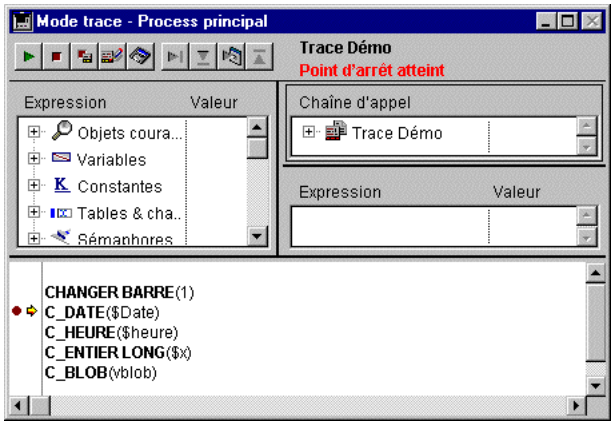


• Cliquer sur le bouton **Tracer** en mode Structure, lorsqu'un process est sélectionné dans la page **Process** de l'Explorateur d'exécution.
• Créer ou modifier un point d'arrêt dans la fenêtre d'édition d'une méthode, ou dans les pages **Point d'arrêt** et **Arrêt sur commande** de l'Explorateur d'exécution.

Note : Lorsque vous tracez un process en cours d'exécution, la fenêtre du débogueur s'affiche instantanément. Si vous tracez un process qui n'est pas en cours d'exécution (process endormi, en attente d'événement, etc...), le débogueur "enregistre" la requête et n'apparaîtra qu'au moment où le process aura repris l'exécution du code.

Note : Si votre base de données est dotée d'un système de mots de passe, seuls le Super_Utilisateur et les utilisateurs possédant les privilèges d'accès à la structure peuvent tracer des méthodes.

Voici la fenêtre du débogueur :

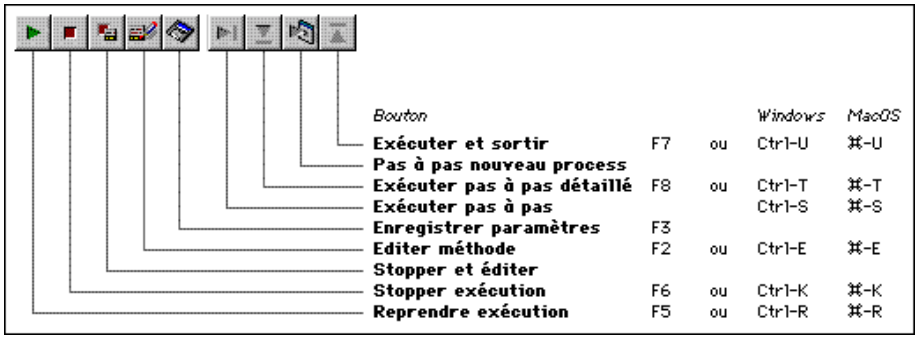


Vous pouvez déplacer la fenêtre du débogueur et/ou redimensionner toutes les zones internes de la fenêtre comme vous le souhaitez. L'affichage ultérieur d'une nouvelle fenêtre du débogueur reprend la configuration (taille et position de la fenêtre, emplacement des lignes de division et contenu de la zone d'évaluation des expressions) de la dernière fenêtre affichée dans la même session.

4D est un environnement multitâche. Dans le cas de plusieurs process s'exécutant simultanément, vous pouvez tracer chacun d'entre eux de manière indépendante. Chaque process peut avoir sa propre fenêtre de débogueur.

Barre d'outils de contrôle d'exécution

La **barre d'outils de contrôle d'exécution**, située en haut de la fenêtre du débogueur, comporte neuf boutons. Voici leur description ainsi que leurs raccourcis clavier associés :



Bouton 'Reprendre exécution'

Arrêt du mode Trace et reprise du cours normal de l'exécution de la méthode.

Note : La combinaison **Maj**+**F5** ou **Maj**+**clic** sur le bouton **Reprendre exécution** provoque la reprise de l'exécution avec désactivation de tous les appels à TRACE suivants dans le process courant.

Bouton 'Stopper exécution'

La méthode s'arrête et vous retournez là où vous étiez avant son exécution. Si vous étiez en train de tracer une méthode formulaire ou une méthode objet s'exécutant en réponse à un événement, elle s'arrête et vous retournez au formulaire. Si vous traciez une méthode s'exécutant à partir du mode Menus créés, vous retournez à ce mode.

Bouton 'Stopper et éditer'

La méthode s'arrête comme lorsque vous cliquez sur **Stopper exécution**. De plus, 4e Dimension ouvre (si nécessaire) le process de Structure, le passe au premier plan et affiche dans la fenêtre de l'éditeur de méthodes la méthode qui s'exécutait au moment où vous avez cliqué sur le bouton.

Conseil : Utilisez ce bouton lorsque vous connaissez les modifications à apporter à votre code, et le moment où elles doivent être effectuées pour pouvoir poursuivre le test de vos méthodes. Une fois vos modifications effectuées, ré-exécutez la méthode.

Bouton 'Editer méthode'

Ce bouton se comporte comme le bouton **Stopper et éditer**, à la différence près qu'il n'annule pas l'exécution en cours. L'exécution de la méthode est simplement suspendue à l'instant du clic sur le bouton. 4e Dimension ouvre (si nécessaire) le process de Structure et le passe au premier plan, puis affiche dans la fenêtre de l'éditeur de méthodes la méthode qui s'exécutait au moment où vous avez cliqué sur le bouton **Editer méthode**.

Important : Vous pouvez modifier cette méthode, mais ces modifications n'apparaîtront pas, ou ne s'exécuteront pas dans l'instance de la méthode tracée dans la fenêtre du débogueur. Ce n'est qu'une fois que la méthode aura été stoppée ou entièrement exécutée, que les modifications pourront apparaître. En d'autres termes, il faut recharger la méthode pour que les modifications soient prises en compte.

Conseil : Utilisez ce bouton lorsque vous connaissez les modifications à apporter à votre code et lorsqu'elles n'interfèrent pas avec le reste du code qui doit être exécuté ou tracé.

Astuce : Les méthodes objet sont rechargées pour chaque événement. Si vous tracez une méthode objet (par exemple en réponse à un clic sur un bouton) vous n'avez pas besoin de quitter le formulaire. Vous pouvez modifier la méthode objet, sauvegarder les modifications, puis retourner au formulaire et tester à nouveau. Pour tracer/modifier les méthodes formulaire, il faut sortir du formulaire puis le réouvrir pour recharger la méthode correspondante. L'astuce est donc la suivante : lorsque vous effectuez le débogage complet d'un formulaire, placez le code que vous déboguez dans une méthode projet que vous utiliserez comme sous-routine à l'intérieur de la méthode formulaire. De cette manière, vous pouvez rester dans le formulaire, tracer, modifier et tester à nouveau votre formulaire car la sous-routine sera rechargée chaque fois qu'elle sera appelée par la méthode formulaire.

Bouton 'Enregistrer paramètres'

Ce bouton permet de sauvegarder la configuration courante de la fenêtre du débogueur (taille et position de la fenêtre, emplacement des lignes de division et contenu de la zone d'évaluation des expressions). Elle sera alors utilisée par défaut à chaque ouverture de la base.
Ces paramétrages sont stockés dans le fichier de structure de la base.

Bouton 'Exécuter pas à pas'

La ligne courante de la méthode (indiquée par la flèche jaune — cette flèche s'appelle le **compteur de programme**) est exécutée et le débogueur passe à la ligne suivante. Le bouton **Exécuter pas à pas** ne passe pas dans les sous-routines et les fonctions. Il reste au niveau de la méthode que vous êtes en train de tracer. Si vous voulez également tracer les appels aux sous-routines et aux fonctions, utilisez le bouton **Pas à pas détaillé**.

Bouton 'Pas à pas détaillé'

Lors de l'exécution d'une ligne qui appelle une autre méthode (sous-routine ou fonction), ce bouton provoque l'affichage de la méthode appelée dans la fenêtre du débogueur, et permet au développeur de passer pas à pas dans cette méthode. La nouvelle méthode devient la méthode courante (en haut) dans la sous-fenêtre Chaîne d'appel de la fenêtre du débogueur. Lors de l'exécution d'une ligne qui n'appelle pas une autre méthode, ce bouton se comporte comme le bouton **Exécuter pas à pas**.

Bouton 'Pas à pas nouveau process'

Lors de l'exécution d'une ligne qui crée un nouveau process (par exemple qui appelle la commande Nouveau process) ce bouton ouvre une nouvelle fenêtre du débogueur qui vous permet de tracer la méthode de gestion du process que vous venez de créer. Lors de l'exécution d'une ligne qui ne crée pas de nouveau process, ce bouton se comporte comme le bouton **Exécuter pas à pas**.

Bouton 'Exécuter et sortir'

Si vous tracez des sous-routines et des fonctions, cliquer sur ce bouton vous permet d'exécuter l'intégralité de la méthode qui est en train d'être tracée, et de revenir à la méthode appelante. La fenêtre du débogueur retourne à la méthode précédente dans la chaîne d'appel. Si la méthode courante est la dernière méthode de la chaîne d'appel, la fenêtre du débogueur se referme.

Informations dans la barre d'outils de contrôle d'exécution

A la droite de la barre d'outils de contrôle d'exécution, le débogueur affiche les informations suivantes :
• Le nom de la méthode que vous êtes en train de tracer (en noir).
• La cause de l'ouverture du débogueur (en rouge).

Par exemple, dans la fenêtre affichée en tête de ce chapitre, vous pouvez voir les informations suivantes :
• La méthode Trace démo est la méthode qui est tracée.
• La fenêtre du débogueur s'affiche car il a rencontré un point d'arrêt.

Voici les causes qui provoquent l'apparition du débogueur et d'un message (en rouge) :
• **Commande TRACE** : un appel à TRACE a été émis.
• **Point d'arrêt atteint** : vous avez rencontré un point d'arrêt temporaire ou persistant.
• **Interruption demandée par l'utilisateur** : vous avez activé **Alt**+**Maj**+**clic droit** (sous Windows) ou **Control**+**Option**+**Commande**+**clic** (sous MacOS) ou encore cliqué sur le bouton **Trace** dans la page **Process** de l'Explorateur d'exécution.
• **Détection d'un appel à : Nom de la commande** : Un appel à une commande 4D qui doit être interceptée est sur le point d'être exécuté.
• **Pas à pas nouveau process** : Vous avez cliqué sur le bouton **Pas à pas nouveau process** et ce message est affiché par la fenêtre du débogueur ouverte pour le process qui vient d'être créé.

Les sous-fenêtres du débogueur

La fenêtre du débogueur contient la barre de contrôle d'exécution décrite précédemment, ainsi que quatre sous-fenêtres redimensionnables :

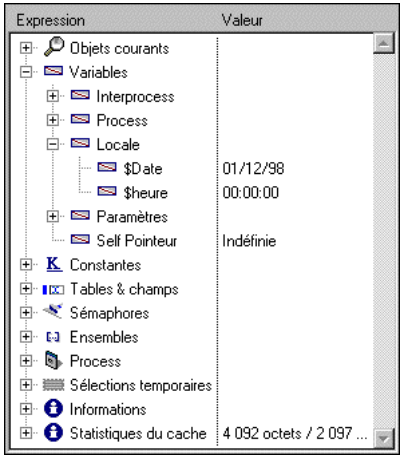
- La Fenêtre d'expression
- La Fenêtre de chaîne d'appel
- La Fenêtre d'évaluation
- La Fenêtre d'évaluation des méthodes

Les trois premières fenêtres affichent des listes hiérarchiques montrant l'information pertinente au débogage. La quatrième, la Fenêtre d'évaluation des méthodes, affiche le code source de la méthode tracée. Chaque fenêtre a un rôle précis pour vous assister dans le processus de débogage. Avec la souris, vous pouvez redimensionner la fenêtre du débogueur, ainsi que chaque sous-fenêtre. En outre, les trois premières sous-fenêtres sont séparées par une ligne pointillée dans le sens de la hauteur : vous pouvez redimensionner à votre convenance les colonnes à l'intérieur de chaque sous-fenêtre.

Référence

APPELER SUR ERREUR, Fenêtre d'erreur de syntaxe, Fenêtre d'évaluation, Fenêtre d'évaluation des méthodes, Fenêtre d'expression, Fenêtre de chaîne d'appel, Liste des points d'arrêt, Points d'arrêt sur commandes, Raccourcis du débogueur, TRACE, Un débogueur, pour quoi faire ?.

La **Fenêtre d'expression** est située en haut à gauche de la fenêtre du débogueur, sous la barre d'outils de contrôle d'exécution :



La Fenêtre d'expression affiche toutes les informations générales utiles sur l'environnement système, 4D et l'exécution du code.

La colonne **Expression** affiche le nom des objets et des expressions. La colonne **Valeur** affiche la valeur courante correspondant aux objets et expressions.

En cliquant sur une valeur dans la colonne de droite, vous pouvez modifier la valeur de l'objet, si cela est possible pour cet objet.

La liste hiérarchique multi-niveaux est organisée par thèmes au niveau supérieur. Les thèmes sont les suivants :

- Objets courants
- Variables
- Constantes
- Tables & champs
- Sémaphores
- Ensembles
- Process
- Sélections temporaires
- Informations
- Statistiques du cache

En fonction du thème, chaque article peut disposer d'un ou de plusieurs sous-niveaux. Pour déployer ou fermer chaque thème, il suffit de cliquer sur l'icône de déploiement située en face de son libellé. Si le thème comprend plusieurs niveaux d'information, il suffit de cliquer sur chaque icône pour explorer toutes les informations qu'il contient.

A tout moment, vous pouvez faire glisser un thème, un sous-thème ou un article, et le déposer dans la Fenêtre d'évaluation.

Objets courants

Ce thème affiche les valeurs des objets ou des expressions évaluables :
• utilisé(e)s dans la ligne de code à exécuter (celle qui est indiquée par le compteur de programme — la flèche jaune dans la fenêtre d'évaluation des méthodes),
• utilisé(e)s dans la ligne de code précédente.

Comme la ligne de code précédente est celle qui vient d'être exécutée, le thème Objets courants affiche donc de manière permanente les objets ou expressions de la ligne courante **avant et après** l'exécution de la ligne. Prenons l'exécution de la méthode suivante :

```
TRACE
a:=1
b:=a+1
c:=a+b
   ` ...
```

1. Vous entrez dans la fenêtre du débogueur avec le compteur de programme de la Fenêtre d'évaluation des méthodes placé à la ligne a:=1. A ce point précis, le thème Objets courants affiche :

a: Indéfini

La variable a est affichée car elle est utilisée dans la ligne qui est sur le point d'être exécutée (mais elle n'a pas encore été initialisée).

2. Vous progressez d'une ligne. Le compteur de programme est maintenant positionné sur la ligne b:=a+1. A ce point, le thème Objets courants affiche :

a: 1
b: Indéfini

La variable a s'affiche car elle est utilisée dans la ligne qui vient de s'exécuter, et qu'on lui avait assigné la valeur numérique 1. Elle s'affiche aussi parce qu'elle est utilisée dans la ligne qui sera exécutée, en tant qu'expression qui sera assignée à la variable b. La variable b s'affiche car elle est utilisée dans la ligne qui doit être exécutée (mais elle n'a pas encore été initialisée).

3. Vous progressez encore d'une ligne. Le compteur de programme est maintenant positionné sur la ligne c:=a+b. A ce point, le thème Objets courants affiche :

```
c:    Indéfini
a:    1
b:    2
```

La variable c s'affiche car elle est utilisée dans la ligne à exécuter (mais elle n'a pas encore été initialisée). Les variables a et b sont affichées car elles étaient utilisées dans la ligne précédente et qu'elles le sont dans la ligne qui sera exécutée, etc.

Ce thème est extrêment pratique : chaque fois que vous exécutez une ligne, vous n'avez pas besoin de saisir une expression dans la fenêtre d'évaluation. Il vous suffit de surveiller les valeurs affichées par le thème Objets courants.

Variables

Ce thème se décompose en sous-thèmes :

• **Interprocess** : affiche la liste des variables interprocess en cours d'utilisation. Si vous n'utilisez pas de variable interprocess, la liste est vide. La valeur des variables interprocess peut être modifiée.

• **Process** : affiche la liste des variables process utilisées par le process courant. Cette liste est rarement vide. La valeur des variables process peut être modifiée.

• **Locales** : affiche la liste des variables locales utilisées par la méthode indiquée dans la sous-fenêtre d'évaluation de méthode. Cette liste peut être vide si aucune variable locale n'est utilisée ou si elles n'ont pas encore été créées. La valeur des variables locales peut être modifiée.

• **Paramètres** : affiche la liste des paramètres reçus par la méthode. Cette liste peut être vide si aucun paramètre n'a été passé à la méthode tracée. La valeur des paramètres peut être modifiée.

• **Self Pointeur** : affiche un pointeur vers l'objet courant si vous tracez une méthode objet. Cette valeur n'est pas modifiable.

Note : Vous pouvez modifier les variables et les champs de type Chaîne, Texte, numérique (Entier, Entier long, Réel), Date et Heure, c'est-à-dire les variables et les champs dont la valeur peut être saisie au clavier. Pour le type Entier vous pouvez utiliser indifféremment la notation décimale ou hexadécimale (par exemple, pour 256, vous pouvez taper '256' ou '0x100').

Les tableaux, comme les autres variables, apparaissent dans les sous-thèmes Interprocess, Process et Locales, en fonction de leur portée. Le débogueur affiche chaque tableau avec un niveau hiérarchique supplémentaire, ce qui vous permet d'obtenir ou de modifier les valeurs des éléments du tableau, s'il y en a. Le débogueur affiche les 100 premiers éléments (y compris l'élément zéro). La colonne Valeur affiche la taille du tableau en face de son nom. Une fois que vous avez déployé le tableau, le premier sous-article affiche le numéro de l'élément sélectionné courant, ensuite l'élément zéro, ensuite les autres éléments (jusqu'à 100) s'il y en a. Vous pouvez modifier les tableaux Alpha, Texte, Numérique et Date. Vous pouvez modifier le numéro de l'élément sélectionné, l'élément zéro et les autres éléments (jusqu'à 100) s'il y en a. Vous ne pouvez pas modifier la taille du tableau.

Note : A tout moment, vous pouvez glisser un article à partir de la Fenêtre d'expression vers la Fenêtre d'évaluation, y compris un élément de tableau.

Constantes : Ce thème affiche les constantes prédéfinies dans 4D, comme dans la page Constantes de la fenêtre de l'Explorateur. Les expressions de ce thème ne peuvent pas être modifiées.

Tables & champs : Ce thème affiche la liste des tables et des champs dans la base de données (à l'exception des sous-champs). Pour chaque table, la colonne Valeur affiche la taille de la sélection courante pour le process courant ainsi que (lorsque la ligne de la table est déployée) le nombre d'enregistrements verrouillés. Pour chaque champ, la colonne Valeur affiche la valeur du champ (à l'exception des images, sous-tables et BLOBs) pour l'enregistrement courant, s'il existe. Dans ce thème, les valeurs des champs peuvent être modifiées (notez qu'il n'y a alors pas d'annulation possible) mais pas celles de la table.

Sémaphores : Ce thème affiche la liste des sémaphores locaux dans les ensembles courants. Pour chaque sémaphore, la colonne Valeurs affiche le nom du process ayant posé le sémaphore. Si vous n'utilisez pas de sémaphore, la liste peut être vide. Les expressions de ce thème ne peuvent pas être modifiées. Les sémaphores globaux ne peuvent pas être visualisés.

Ensembles : Ce thème affiche la liste des ensembles définis dans le process courant (celui que vous êtes en train de tracer) ainsi que la liste des ensembles interprocess. La colonne Valeur affiche, pour chaque ensemble, le nombre d'enregistrements et le nom de la table. Si vous n'utilisez pas les ensembles, cette liste sera vide. Les expressions de ce thème ne peuvent pas être modifiées.

Process : Ce thème affiche la liste des process lancés depuis le début de la session de travail. La colonne Valeur affiche le temps déjà alloué à chaque process ainsi que son état (par exemple "En cours d'exécution", "Endormi", etc). Les expressions de ce thème ne peuvent pas être modifiées.

Sélections temporaires : Ce thème affiche la liste des sélections temporaires process définies dans le process courant (celui que vous êtes en train de tracer) ainsi que la liste des sélections temporaires interprocess. Pour chaque sélection temporaire, la colonne Valeur affiche le nombre d'enregistrements et le nom de la table. Si vous n'utilisez pas les sélections temporaires, cette liste sera vide. Les expressions de ce thème ne peuvent pas être modifiées.

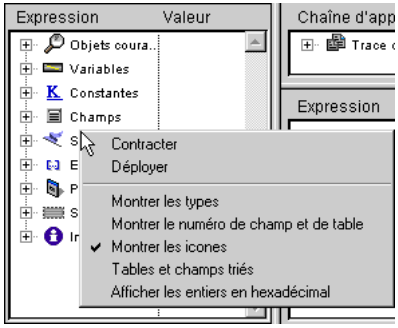
Informations : Ce thème affiche les informations générales, comme la table par défaut courante (s'il y en a une). Les expressions contenues dans ce thème ne peuvent pas être modifiées.

Statistiques du cache : Ce thème affiche diverses statistiques concernant l'utilisation des tables d'adresses et d'index, des pages d'index et des sélections temporaires chargées dans le cache de 4D. Les expressions contenues dans ce thème ne peuvent pas être modifiées.

Menu contextuel de la fenêtre d'expression

Le menu contextuel de la fenêtre d'expression vous propose des options supplémentaires. Pour afficher ce menu il vous suffit de :
• Sous Windows, cliquer n'importe où dans la fenêtre d'expression avec le **bouton droit** de la souris
• Sous MacOS, utiliser la combinaison **Control**+**clic** n'importe où dans la fenêtre d'expression.

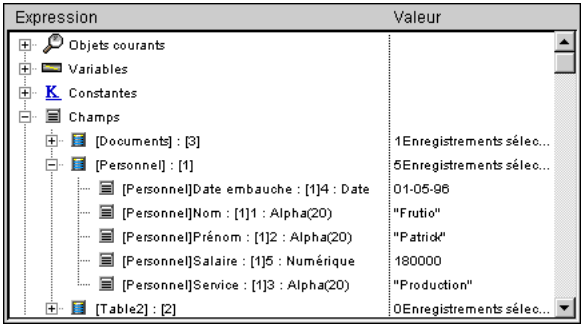
Voici le menu contextuel de la fenêtre d'expression :



• **Contracter** : Contracte tous les niveaux de la liste hiérarchique des expressions.
• **Déployer** : Déploie tous les niveaux de la liste hiérarchique des expressions.
• **Montrer les types** : Lorsque vous sélectionnez cette option, le type de l'objet s'affiche en face de l'objet (lorsque cela est pertinent).

• **Montrer le numéro de champ et de table** : Si vous travaillez avec le numéro des tables ou de champs, ou avec des pointeurs utilisant les commandes Table ou Champ, cette option est très pratique : en face de chaque table et champ, elle affiche le numéro de la table ou du champ.
• **Montrer les icônes** : Chaque objet est précédé d'une icône qui indique son type. Vous pouvez désactiver cette option pour accélérer l'affichage, ou tout simplement parce que l'option **Montrer les types** vous convient.
• **Tables et champs triés** : Cette option force les tables et les champs à s'afficher par ordre alphabétique (dans leurs listes respectives).
• **Afficher les entiers en hexadécimal** : Les nombres s'affichent en notation décimale. Sélectionnez cette option pour les afficher en hexadécimal.

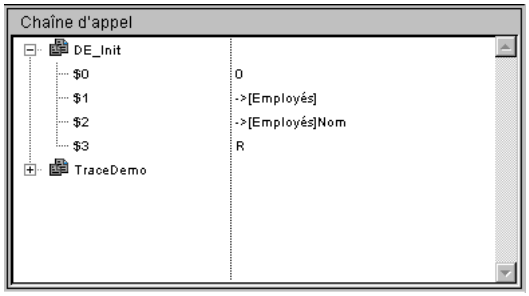
Ci-dessous, la fenêtre d'expression telle qu'elle se présente lorsque vous sélectionnez toutes les options :



Référence

Débogueur, Fenêtre d'évaluation, Fenêtre d'évaluation des méthodes, Fenêtre de chaîne d'appel, Raccourcis du débogueur.

Une méthode peut appeler d'autres méthodes, qui à leur tour peuvent appeler d'autres méthodes. Pour cette raison, il est très utile d'avoir sous les yeux, pendant le débogage, la chaîne des méthodes, ou **chaîne d'appel**. Cette chaîne peut être visualisée dans la fenêtre située en haut et à droite du débogueur. Les méthodes y sont affichées de manière hiérarchique :



• Chaque niveau principal est le nom d'une méthode. L'élément placé en tête de la liste est la méthode que vous êtes en train de tracer, le niveau suivant est le nom de la méthode appelante (la méthode qui a appelé celle que vous êtes en train de tracer), le niveau suivant est l'appelant de la méthode appelante, etc. Dans l'exemple ci-dessus, la méthode DE_Init est tracée. Elle a été appelée par la méthode TraceDemo.

• Lorsque vous double-cliquez sur le nom d'une méthode dans la fenêtre de chaîne d'appel, vous basculez sur la méthode appelante dont le code source est affiché dans la fenêtre d'évaluation de méthodes. Vous pouvez ainsi voir rapidement comment la méthode appelante a effectué son appel à la méthode appelée. Vous pouvez aussi examiner toutes les étapes de la chaîne d'appel.

• Lorsque vous cliquez sur l'icône de déploiement jouxtant le nom d'une méthode, vous déployez ou vous contractez la liste des paramètres ($1, $2...) ainsi que le résultat ($0) optionnel d'une fonction. La valeur s'affiche à droite de la fenêtre. En cliquant sur une valeur quelconque à droite, vous pouvez changer la valeur du résultat ou de tout paramètre.

Dans l’illustration ci-dessus :

1. DE_Init $0 est actuellement indéfinie car la méthode n’a assigné aucune valeur à $0 (parce qu'elle n’a pas encore exécuté cette affectation, ou parce que la méthode est une sous-routine et non une fonction).
2. DE_Init a reçu trois paramètres de TraceDemo. $1 est un pointeur vers la table [Employés], $2 est un pointeur vers le champ [Employés]Nom et $3 est un paramètre alphanumérique de valeur “R”.

• Lorsque vous avez déployé la liste des paramètres/résultats d’une méthode, vous pouvez également les faire glisser vers la Fenêtre d’évaluation.

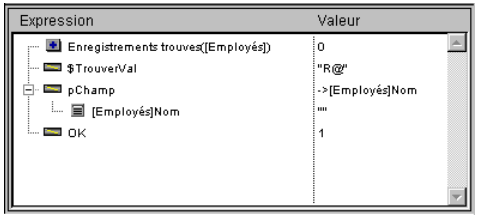
Référence

Débogueur, Fenêtre d'évaluation, Fenêtre d'évaluation des méthodes, Fenêtre d'expression, Raccourcis du débogueur.

Juste au-dessous de la Fenêtre de chaîne d'appel se trouve la Fenêtre d'évaluation. Cette fenêtre sert à évaluer les expressions. Vous pouvez évaluer tout type d'expression, y compris les champs, les variables, les pointeurs, les calculs, les fonctions intégrées, vos propres fonctions, et tout ce qui retourne une valeur.

Vous pouvez évaluer toute expression qui peut être affichée sous forme de texte. Cela ne s'applique donc pas aux champs ni aux variables image ou BLOB. En revanche, lorsque vous déployez les listes hiérarchiques, le débogueur vous permet d'afficher les tableaux et les pointeurs. Pour afficher le contenu des BLOBs, vous pouvez utiliser les commandes sur les BLOBs, comme par exemple BLOB vers texte.

Dans l'exemple ci-dessous, vous pouvez voir plusieurs expressions : deux variables, un pointeur vers un champ, le résultat d'une fonction intégrée et un calcul.



Insérer une nouvelle expression

Vous pouvez ajouter une expression dans la fenêtre de l'une des manières suivantes :
- Glissez-déposez un objet ou une expression à partir de la Fenêtre d'expression ;
- Glissez-déposez un objet ou une expression à partir de la Fenêtre de chaîne d'appel ;
- Cliquez sur une expression évaluable dans la Fenêtre d'évaluation des méthodes.

En outre, pour créer une expression vide, double-cliquez n'importe où dans l'espace vide de la fenêtre d'évaluation. Cela crée une expression vide ` Nouvelle expression prête à l'édition. Vous pouvez saisir toute formule 4D qui retourne un résultat.

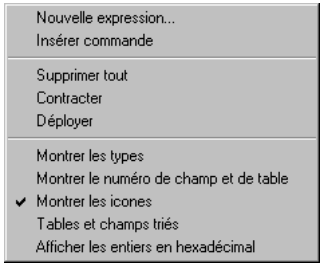
Dès que vous avez saisi la formule, appuyez sur la touche **Entrée** ou **Retour chariot** (ou cliquez n'importe où dans la fenêtre) pour obtenir l'évaluation de l'expression.

Si vous voulez changer d'expression, cliquez dessus pour la sélectionner, et cliquez de nouveau pour entrer en mode édition.

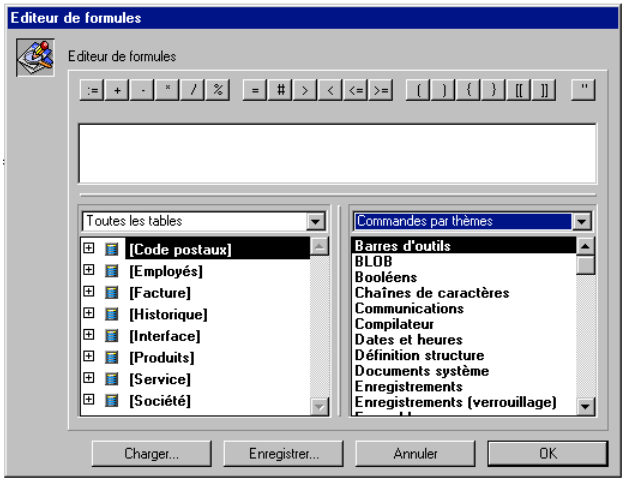
Si vous n'avez plus besoin d'une expression, cliquez dessus pour la sélectionner, puis appuyez sur la touche **Retour Arrière** ou **Suppr.**

Menu contextuel de la fenêtre d'évaluation

Le menu contextuel de la Fenêtre d'évaluation vous permet d'accéder à l'éditeur de formules de 4D, pour vous aider à saisir et éditer une expression. Le menu contextuel vous propose également des options supplémentaires. Pour obtenir ce menu :
• Sous Windows, cliquez n'importe où dans la fenêtre d'évaluation avec le **bouton droit** de la souris.
• Sous MacOS, utilisez la combinaison **Control**+**clic** n'importe où dans la fenêtre d'évaluation.



• **Nouvelle expression** : Cette commande insère une nouvelle expression et affiche l'éditeur de formules de 4D (voir ci-dessous) dans lequel vous pouvez saisir la nouvelle expression.



Pour plus d'informations sur l'éditeur de formules, reportez-vous au manuel *Mode Utilisation* de 4D.

• **Insérer commande** : Cette commande est un raccourci pour placer une commande (sans utiliser l'éditeur de formules) en tant que nouvelle expression.
• **Supprimer tout** : Efface toutes les expressions présentes.
• **Contracter/Déployer** : Contracte ou déploie toutes les expressions dont l'évaluation se fait en utilisant les listes hiérarchiques (pointeurs, tableaux, etc.).
• **Montrer les types** : Si cette option est sélectionnée, le type de l'objet s'affiche en face de chaque objet (lorsque c'est pertinent).
• **Montrer le numéro de champ et de table** : Cette commande est extrêmement pratique si vous travaillez avec des numéros de table ou de champs, ou avec des pointeurs utilisant des commandes comme Table ou Champ. En face de chaque table et champ, elle affiche le numéro de la table ou du champ.
• **Montrer les icônes** : Chaque objet est précédé par une icône qui symbolise son type. Vous pouvez désactiver cette option pour accélérer l'affichage, ou simplement parce que l'option **Montrer les types** vous convient.
• **Tables et champs triés** : Cette option force l'affichage des tables et des champs par ordre alphabétique (dans leurs listes respectives).
• **Afficher les entiers en hexadécimal** : Par défaut, les nombres entiers sont affichés en notation décimale. Sélectionnez cette option si vous souhaitez un affichage hexadécimal.

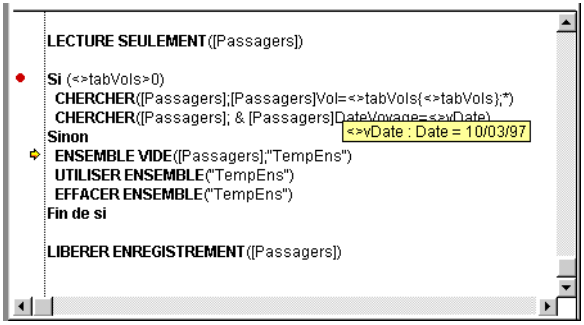
Référence

Débogueur, Fenêtre d'évaluation des méthodes, Fenêtre d'expression, Fenêtre de chaîne d'appel, Raccourcis du débogueur.

La Fenêtre d'évaluation des méthodes affiche le code source de la méthode en train d'être tracée.

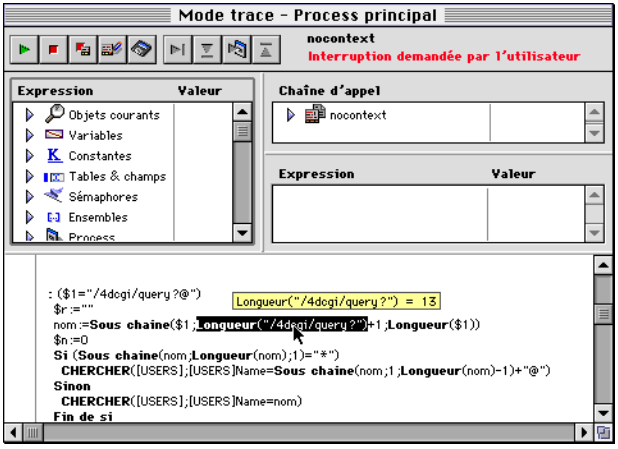
• Si la méthode est trop longue pour tenir dans la zone de texte, vous pouvez la faire défiler.

• Lorsque vous positionnez le pointeur de la souris au-dessus d'une expression qui peut être évaluée (champ, variable, pointeur, tableau,...) le débogueur affiche une **Info-bulle** qui indique la valeur courante de l'objet ou de l'expression, et son type déclaré. Par exemple :



Une info-bulle apparaît lorsque le pointeur de la souris est positionné sur la variable <>vDate. Dans cet exemple, c'est une variable de type date contenant la valeur suivante : 10/03/97.

• Vous pouvez également sélectionner une portion de texte dans la zone affichant le code en cours d'exécution. Dans ce cas, lorsque le curseur de la souris est placé au-dessus du texte sélectionné, l'info-bulle fournit la valeur de la sélection :



Lorsque vous cliquez sur un nom de variable ou de champ, il est automatiquement sélectionné.

Astuce : Vous pouvez copier instantanément dans la Fenêtre d'évaluation l'expression ou l'objet sélectionné. Vous disposez de trois solutions :
• par simple glisser-déposer : cliquez sur le texte sélectionné, faites-le glisser et relâchez-le dans la zone d'évaluation.
• cliquer sur le texte sélectionné tout en maintenant enfoncée la touche **Ctrl** (Windows) ou **Commande** (MacOS).
• utiliser la combinaison de touches **Ctrl+D** (Windows) ou **Commande+D** (MacOS).

Compteur de programme

Une flèche jaune, dans la marge gauche de la fenêtre d'évaluation des méthodes (voir illustration ci-dessus) indique la prochaine ligne à être exécutée. Cette flèche s'appelle le **compteur de programme**. Elle indique toujours la ligne sur le point d'être exécutée.

Pendant le débogage, vous pouvez **déplacer** le compteur de programme. Il vous suffit de cliquer sur la flèche jaune et de la faire glisser devant la ligne que vous désirez.

ATTENTION : Utilisez cette fonction avec précaution ! Déplacer vers l'avant le compteur de programme ne signifie pas que le débogueur exécute en même temps les lignes sur lesquelles vous passez. De la même manière, le faire remonter ne signifie pas que le débogueur inverse l'effet des lignes qui ont déjà été exécutées.

Lorsque vous déplacez le compteur de programme, vous indiquez simplement au débogueur de "continuer à tracer et exécuter à partir de cet endroit ". Tous les paramètres, champs, variables, etc. courants, ne sont pas affectés par le déplacement.

Voici un exemple de déplacement du compteur de programme. Imaginons que vous êtes en train de déboguer le code suivant :

```
` ...
Si (Cette condition)
   FAIRE QUELQUE CHOSE
Sinon
   FAIRE AUTRE CHOSE
Fin de si
```

Le compteur de programme est placé à la ligne Si(Cette condition). Vous avancez d'un pas, et voyez que le compteur se place à la ligne FAIRE AUTRE CHOSE. Pas de chance, car vous vouliez en fait exécuter l'autre partie de l'alternative. Dans ce cas, et étant donné que l'expression Cette condition n'effectue pas d'opérations affectant les étapes suivantes de votre test, remontez le compteur à la ligne FAIRE QUELQUE CHOSE, et vous êtes prêt à continuer à tracer la portion de code qui vous intéresse.

Définir des points d'arrêt dans le débogueur

Pendant le débogage, vous pouvez avoir besoin de sauter certaines portions de votre code. Le débogueur vous permet d'utiliser plusieurs méthodes pour exécuter votre code **jusqu'à un certain point** :

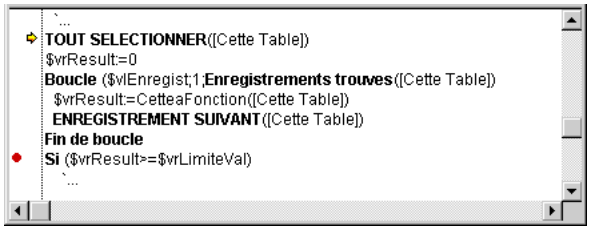
• Pendant que vous exécutez au pas à pas, vous pouvez cliquer sur le bouton **Exécuter pas à pas** au lieu de **Pas à pas détaillé**, lorsque vous ne voulez pas entrer dans les éventuelles sous-routines ou fonctions appelées dans la ligne du compteur de programme.
• Si vous entrez par erreur dans une sous-routine, vous pouvez l'exécuter et revenir directement à la méthode appelante en cliquant sur le bouton **Exécuter et sortir**.
• Si vous appelez la commande TRACE quelque part, vous pouvez cliquer sur le bouton **Pas de Trace** pour reprendre l'exécution jusqu'à cet appel TRACE.

Par exemple, imaginons que vous soyez en train d'exécuter le code suivant. Le compteur de programme est placé devant la ligne TOUT SELECTIONNER ([CetteTable]) :

```
    ` ...
TOUT SELECTIONNER([CetteTable])
$vrResult:=0
Boucle($vlEnregist;1;Enregistrements trouves([CetteTable]))
   $vrResult:=CetteaFonction([CetteTable])
   ENREGISTREMENT SUIVANT([CetteTable])
Fin de boucle
Si ($vrResult>=$vrLimiteVal)
   ` ...
```

Vous voulez évaluer la valeur de $vrResult à la fin de la boucle Boucle. Comme cela peut prendre un certain temps d'exécution pour atteindre cette portion de votre code, vous voulez abandonner l'exécution courante puis éditer la méthode pour insérer un appel TRACE avant la ligne Si ($vrResult...

Vous pourriez aussi exécuter la boucle, mais si la table [CetteTable] contient plusieurs centaines d'enregistrements, cette opération vous prendra la journée. Pour éviter des situations de ce type, le débogueur met à votre disposition des **points d'arrêt**. Vous insérez des points d'arrêt en cliquant dans la marge gauche de la fenêtre d'évaluation des méthodes.
Par exemple, vous cliquez dans la marge gauche de la fenêtre, au niveau de la ligne Si ($vrResult... :



Vous insérez un point d’arrêt à la ligne marquée par un point rouge. Ensuite, vous cliquez sur le bouton **Pas de Trace**.
Vous reprenez l’exécution normale **jusqu’à** la ligne marquée par le point d’arrêt. Cette ligne n’est pas exécutée : vous êtes revenu au mode Trace. Dans cet exemple, toute la boucle a été exécutée normalement, et, une fois arrivé au point d’arrêt, il vous suffit de placer le curseur de la souris au-dessus de vrResult pour évaluer sa valeur à la sortie de la boucle.
Si vous placez un point d’arrêt au-delà du compteur de programme et que vous cliquez sur le bouton **Pas de Trace**, vous éviterez de tracer des parties de la méthode.

Note : Vous pouvez également définir des points d'arrêt directement dans l'éditeur de méthodes de 4D. Reportez-vous à la section Points d'arrêt.

Un **point d’arrêt rouge** est un point d’arrêt **persistant**. Une fois que vous l’avez créé, il reste, même lorsque vous quittez la base et la réouvrez par la suite.

Pour le supprimer, vous pouvez procéder d'une des manières suivantes :
• Si vous avez fini de l’utiliser, cliquez dessus, et le point d’arrêt disparaît.
• Si vous n’avez pas fini de l’utiliser, mais que vous voulez le désactiver provisoirement, il vous faut l’éditer. Reportez-vous pour ceci à la section Points d’arrêt.

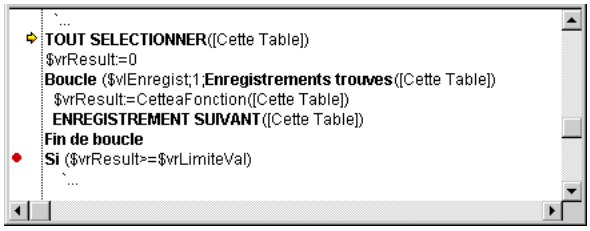
Référence

Débogueur, Fenêtre d'évaluation, Fenêtre d'expression, Fenêtre de chaîne d'appel, Points d'arrêt.

Comme décrit dans la section Fenêtre d'évaluation des méthodes, vous définissez un point d'arrêt en cliquant dans la marge gauche de la fenêtre d'évaluation des méthodes ou de la fenêtre d'édition des méthodes, au niveau de la ligne de code sur laquelle vous voulez vous arrêter.

Note : L'insertion et l'édition de points d'arrêt peuvent être effectuées indifféremment dans l'éditeur de méthodes ou le débogueur. Il y a une interaction dynamique entre les deux éditeurs : un point d'arrêt inséré ou modifié dans un éditeur est immédiatement reporté dans l'autre (ainsi que dans la Liste des points d'arrêt de l'explorateur d'exécution). A noter toutefois que les points d'arrêt **provisoires** ne peuvent être créés que dans le débogueur (cf. ci-dessous).

Dans l'exemple suivant, un point d'arrêt a été placé, dans le débogueur, en regard de la ligne
Si ($vrResult>=$vrLimitVal) :



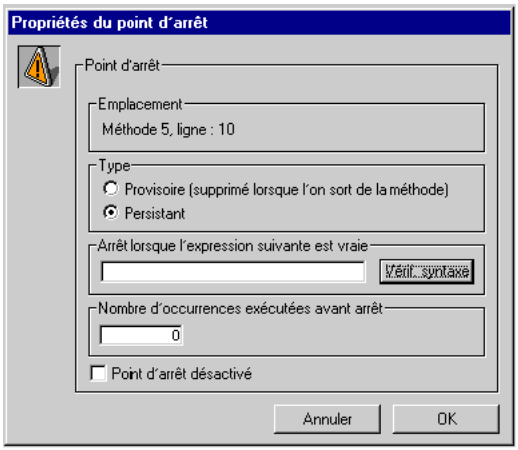
Si vous cliquez de nouveau sur la puce rouge, vous supprimez le point d'arrêt.

Editer un point d'arrêt

La combinaison **Alt**+**clic** (sous Windows) ou **Option**+**clic** (sous MacOS) dans la marge gauche de la fenêtre d'évaluation des méthodes (ou de l'éditeur de méthodes) ouvre la **fenêtre des propriétés du point d'arrêt**.

• Si vous avez cliqué sur un point d'arrêt existant, la fenêtre s'ouvre pour ce point d'arrêt.
• Si vous cliquez sur une ligne où aucun point d'arrêt n'a été placé, un point d'arrêt est créé et sa fenêtre de propriétés s'affiche.

Voici la **fenêtre des propriétés du point d'arrêt** :



• **Emplacement** : Vous indique le nom de la méthode et le numéro de la ligne où le point d'arrêt a été placé. Vous ne pouvez pas modifier cette information.

• **Type** : Par défaut, vous créez des points d'arrêt **persistants**, symbolisés par une puce rouge. Toutefois, le débogueur vous permet de définir des **points d'arrêt provisoires**. Il est utile de pouvoir disposer de points d'arrêt provisoires lorsque vous voulez vous arrêter une seule fois dans une méthode. Un point d'arrêt provisoire est identifié dans la fenêtre d'évaluation des méthodes du débogueur par une **puce verte**. Vous pouvez poser un point d'arrêt provisoire directement dans la fenêtre d'évaluation des méthodes en cliquant dans la marge gauche tout en maintenant les touches **Alt** + **Majuscule** (Windows) ou **Option** + **Majuscule** (Macintosh) enfoncées.

Note : Les points d'arrêt provisoires peuvent être définis dans le débogueur uniquement.

• **Arrêt lorsque l'expression suivante est vraie** : Saisissez une formule 4D qui retourne Vrai ou Faux. Si, par exemple, vous voulez ne vous arrêter à une ligne que lorsque, par exemple, Enregistrements dans selection([aTable])=0, saisissez cette formule et l'arrêt se produira uniquement si vous n'avez aucun enregistrement sélectionné pour la table [aTable] lorsque le débogueur rencontrera la ligne où le point d'arrêt est placé. Si vous n'êtes pas sûr de la syntaxe de votre formule, cliquez sur le bouton **Vérif. Syntaxe**.

• **Nombre d'occurrences exécutées avant arrêt** : Vous pouvez insérer un point d'arrêt sur une ligne de code placée dans une structure en boucle (Tant que, Repeter, Boucle) ou placée dans une sous-routine ou fonction appelée à partir d'une boucle. Si vous savez que le problème que vous cherchez n'arrivera pas avant au moins la 200e itération de la boucle, saisissez 200 et le point d'arrêt s'activera à la 201e itération.

• **Point d’arrêt désactivé** : Vous avez créé un point d’arrêt persistant dont vous n’avez plus besoin pour le moment mais qui pourrait vous servir plus tard. Il vous suffit dans ce cas de le désactiver. Un point d’arrêt désactivé s’affiche sous forme de tiret (-) et non plus de puce (•) à la fois dans le débogueur, dans l'éditeur de méthodes et dans la Liste des points d’arrêt.

Les points d’arrêt peuvent être créés et édités dans la fenêtre du débogueur et l'éditeur de méthodes. Mais vous pouvez également éditer des points d’arrêt existants en affichant la liste des points d’arrêt, dans la page "Point d'arrêt" de l'Explorateur d'exécution. Pour plus d’informations, reportez-vous à la section Liste des points d’arrêt.

Référence

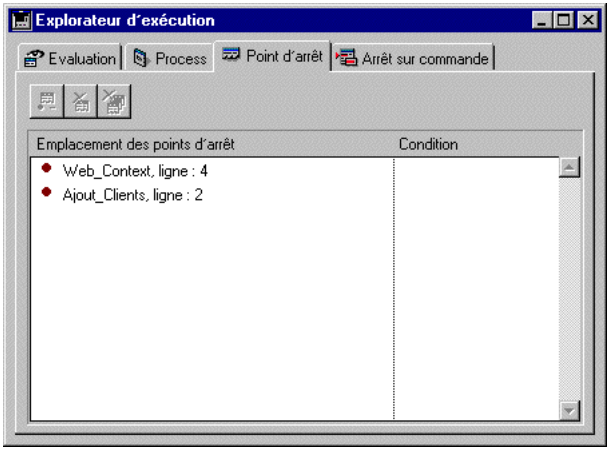
Débogueur, Fenêtre d'évaluation des méthodes, Liste des points d’arrêt, Points d'arrêt sur commandes.

La liste des points d'arrêt est une page de l'Explorateur d'exécution qui vous permet de gérer les points d'arrêt que vous avez créés dans la fenêtre du débogueur ou dans l'éditeur de méthodes.

Pour afficher la Liste des points d'arrêt, procédez de la manière suivante :

1. Passez en mode Structure — si vous n'y êtes pas déjà.
2. Choisissez **Explorateur d'exécution** dans le menu **Outils**.
La fenêtre de l'Explorateur d'exécution peut être affichée sous forme de palette flottante (pour cela, maintenez la touche **Maj** enfoncée au moment de sélectionner la commande de menu). L'Explorateur d'exécution est alors disponible dans tous les environnements de 4D. Pour plus d'informations, reportez-vous au manuel *Mode Structure*.

La fenêtre de l'Explorateur d'exécution apparaît.
3. Cliquez sur l'onglet **Points d'arrêt** pour afficher la liste des points d'arrêt :



La liste des points d'arrêts est divisée en deux colonnes :
• La colonne de gauche indique le statut actif/inactif du point d'arrêt, suivi du nom de la méthode et du numéro de la ligne à laquelle le point d'arrêt a été placé dans la fenêtre du débogueur ou de l'éditeur de méthodes.
• La colonne de droite affiche la condition optionnellement associée au point d'arrêt.

Vous pouvez définir dans cette fenêtre une condition pour un point d'arrêt.
Vous pouvez activer, désactiver ou supprimer chaque point d'arrêt. En revanche, vous ne pouvez pas ajouter de point d'arrêt dans l'Explorateur. Les points d'arrêt persistants ne peuvent être créés qu'à partir de la fenêtre du débogueur ou de l'éditeur de méthodes.
Vous pouvez également ouvrir une fenêtre de l'éditeur de méthodes affichant la méthode à laquelle est associé le point d'arrêt en double-cliquant sur le libellé du point d'arrêt.

Définir une condition pour un point d'arrêt

Pour définir une condition pour un point d'arrêt, procédez de la manière suivante :
1. Cliquez dans la colonne de droite en regard du point d'arrêt.
2. Saisissez une formule 4D (expression ou appel de commande ou méthode projet) retournant une valeur booléenne (Vrai ou Faux).

Note : Pour supprimer une condition, effacez la formule correspondante.

Activer/Désactiver un point d'arrêt

Pour activer ou désactiver un point d'arrêt, procédez de la manière suivante :
1. Sélectionnez le point d'arrêt en cliquant dessus ou en utilisant les touches fléchées pour naviguer dans la liste.
2. Cliquez sur le bouton **Activer/Désactiver** (premier bouton situé au-dessus de la liste) ou sélectionnez la commande correspondante dans le menu contextuel.

Raccourci : Chaque point d'arrêt dans la liste peut être activé/désactivé par un simple clic sur la puce (•) placée devant son libellé. La puce se transforme en tiret (-) lorsque le point d'arrêt est désactivé.

Supprimer un point d'arrêt

Pour supprimer un point d'arrêt, procédez de la manière suivante :
1. Sélectionnez le libellé du point d'arrêt en cliquant dessus ou en utilisant les touches fléchées pour naviguer dans la liste.
2. Appuyez sur la touche **Retour arrière**, ou cliquez sur le bouton **Supprimer** (deuxième bouton situé au-dessus de la liste), ou encore sélectionnez la commande **Supprimer** dans le menu contextuel.

Note : Pour supprimer tous les points d'arrêt, cliquez sur le bouton **Supprimer tout** (troisième bouton situé au-dessus de la liste), ou encore sélectionnez la commande **Supprimer tout** dans le menu contextuel.

Astuces

• L'ajout de conditions aux points d'arrêt persistants ralentit l'exécution du code car la condition est évaluée à chaque fois que le point d'arrêt est rencontré. D'un autre côté, le débogage est plus rapide car 4D ignore automatiquement les occurrences qui ne satisfont pas aux conditions.

• La désactivation d'un point d'arrêt persistant produit quasiment le même effet que sa suppression. Lors de l'exécution du code, le débogueur ne passe presque pas de temps sur le point d'arrêt. L'avantage de la désactivation est que vous n'avez pas à redéfinir le point d'arrêt en cas de besoin.

Référence

Débogueur, Fenêtre d'évaluation des méthodes, Points d'arrêt, Points d'arrêt sur commandes, Un débogueur, pour quoi faire ?.

La liste des Points d'arrêt sur commandes est une page de l'Explorateur d'exécution qui vous permet d'ajouter des points d'arrêt supplémentaires dans votre code en interceptant des appels aux commandes 4D.

Placer un point d'arrêt sur une commande vous permet commencer à tracer l'exécution de n'importe quel process dès qu'une commande particulière est appelée par le process. A la différence d'un point d'arrêt placé dans une méthode projet (qui, par conséquent, déclenche le mode trace uniquement lorsqu'il est atteint), l'aire d'action d'un point d'arrêt sur commande comprend tous les process qui exécutent du code 4D et qui appellent cette commande.

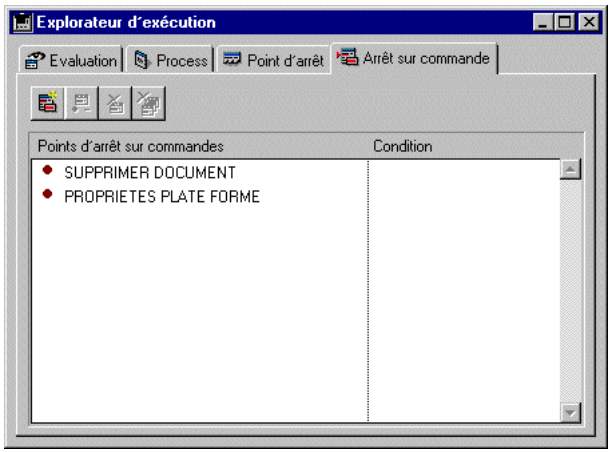
Placer un point d'arrêt sur une commande est un moyen pratique de tracer des grandes parties du code sans devoir insérer des points d'arrêt à des emplacements arbitraires. Si, par exemple, l'exécution dans votre code de plusieurs process durant un certain laps de temps provoque l'effacement d'un enregistrement qui ne devrait théoriquement pas être supprimé, vous pouvez limiter le champ d'investigation en plaçant un point d'arrêt sur les commandes telles que SUPPRIMER ENREGISTREMENT et SUPPRIMER SELECTION. A chaque fois que ces commandes sont appelées, vous pouvez vérifier si l'enregistrement en question a été supprimé ou non, et donc isoler la partie fautive du code.
Bien entendu, l'expérience aidant, vous pourrez combiner l'utilisation de points d'arrêt dans les méthodes et sur des commandes.

Pour afficher la liste des Points d'arrêt sur commandes, procédez de la manière suivante :

1. Passez en mode Structure — si vous n'y êtes pas déjà.
2. Choisissez **Explorateur d'exécution** dans le menu **Outils**.
La fenêtre de l'Explorateur d'exécution peut être affichée sous forme de palette flottante (pour cela, maintenez la touche **Maj** enfoncée au moment de sélectionner la commande de menu). L'Explorateur d'exécution est alors disponible dans tous les environnements de 4D. Pour plus d'informations, reportez-vous au manuel *Mode Structure*.

La fenêtre de l'Explorateur d'exécution apparaît.

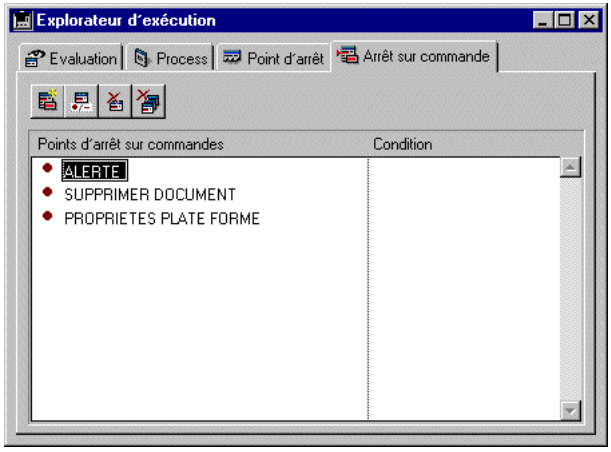
3. Cliquez sur l'onglet **Arrêt sur commande** pour afficher la liste des points d'arrêt sur commande :



Cette page liste les commandes à intercepter au moment de leur exécution. Elle divisée en deux colonnes :
• La colonne de gauche indique le statut actif/inactif du point d'arrêt, suivi du nom de la commande.
• La colonne de droite affiche la condition optionnellement associée au point d'arrêt.

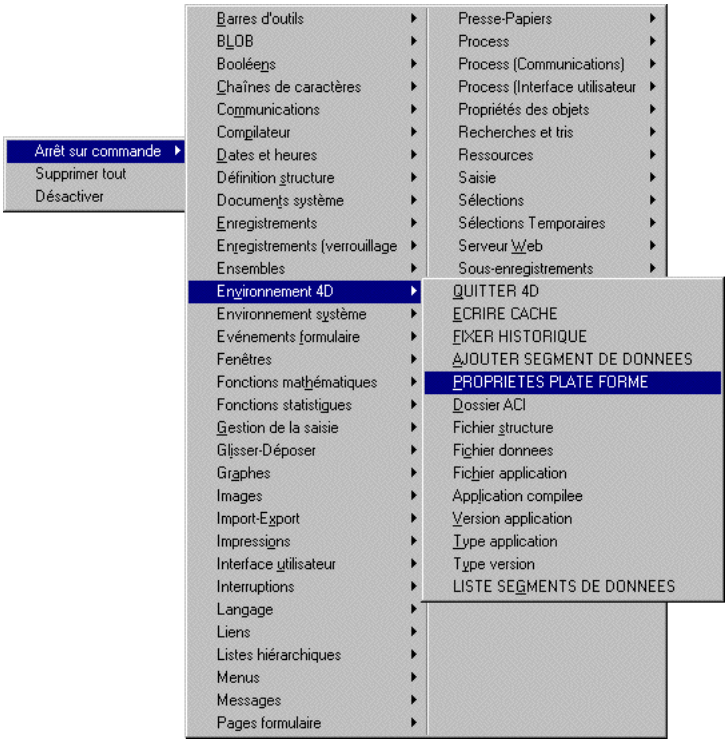
Ajouter un nouveau point d'arrêt sur commande

Pour ajouter un nouveau point d'arrêt sur une commande, vous disposez de plusieurs possibilités :
• cliquer sur le bouton **Arrêt sur commande** (premier bouton placé au-dessus de la liste).
• double-cliquer directement dans la liste.
Dans ces deux cas, une nouvelle entrée est ajoutée dans la liste, avec la commande ALERTE par défaut.



Le libellé ALERTE est automatiquement placé en mode édition, vous pouvez alors saisir le nom de la commande à laquelle vous souhaitez associer un point d'arrêt. Une fois que vous avez terminé, appuyez sur la touche **Entrée** ou **Retour chariot** pour valider votre choix.

• cliquer dans la liste avec le **bouton droit** de la souris (sous Windows) ou effectuer **Control+clic** (sous MacOS) pour afficher un menu contextuel. Sélectionnez **Arrêt sur commande**, le menu liste alors toutes les commandes 4D :



Choisissez votre commande parmi les sous-menus des thèmes puis des noms de commandes. Une nouvelle entrée est ajoutée, affichant le nom de la commande sélectionnée.

Modifier un point d'arrêt sur commande

Pour modifier un point d'arrêt sur une commande, procédez de la manière suivante :

1. Sélectionnez le point d'arrêt en cliquant dessus ou en utilisant les touches fléchées pour naviguer dans la liste (s'il n'y pas déjà un point d'arrêt sélectionné et en mode édition).

2. Pour passer un point d'arrêt du mode sélection au mode édition et inversement, appuyez sur la touche **Entrée** ou **Retour chariot**.

3. Saisissez ou modifiez le nom de la commande.

4. Pour valider vos modifications, appuyez sur la touche **Entrée** ou **Retour chariot**. Si le nom que vous avez saisi ne correspond pas à une commande 4D existante, le point d'arrêt affiche la valeur précédente. Dans le cas d'une nouvelle entrée, le libellé ALERTE est réaffiché.

Activer/Désactiver un point d'arrêt sur commande

Pour activer ou désactiver un point d'arrêt sur commande, procédez de la manière suivante :
1. Sélectionnez le point d'arrêt en cliquant dessus ou en utilisant les touches fléchées pour naviguer dans la liste (s'il n'y pas déjà un point d'arrêt sélectionné et en mode édition).
2. Cliquez sur le bouton **Activer/Désactiver** (deuxième bouton placé au-dessus de la liste), ou choisissez la commande correspondante dans le menu contextuel.

Raccourci : Chaque point d'arrêt dans la liste peut être activé/désactivé par un simple clic sur la puce (•) placée devant son libellé. La puce se transforme en tiret (–) lorsque le point d'arrêt est désactivé.

Supprimer un point d'arrêt sur commande

Pour supprimer un point d'arrêt sur commande, procédez de la manière suivante :
1. Sélectionnez le point d'arrêt en cliquant dessus ou en utilisant les touches fléchées pour naviguer dans la liste (s'il n'y pas déjà un point d'arrêt sélectionné et en mode édition).
2. Assurez-vous que le point d'arrêt se trouve en mode sélection (appuyez sur la **Entrée** ou **Retour chariot** pour passer alternativement du mode édition au mode sélection)
3. Appuyez sur la touche **Retour arrière** ou cliquez sur le bouton **Supprimer** (troisième bouton placé au-dessus de la liste).

Note : Pour supprimer tous les points d'arrêt, cliquez sur le bouton **Supprimer tout** (quatrième bouton placé au-dessus de la liste) ou choisissez la commande **Supprimer tout** dans le menu contextuel.

Définir une condition pour un point d'arrêt sur commande

Pour définir une condition pour un point d'arrêt sur commande, procédez de la manière suivante :
1. Cliquez dans la colonne de droite en regard du point d'arrêt.
2. Saisissez une formule 4D (expression ou appel de commande ou méthode projet) retournant une valeur booléenne (Vrai ou Faux).

Note : Pour supprimer une condition, effacez la formule correspondante.

Astuces

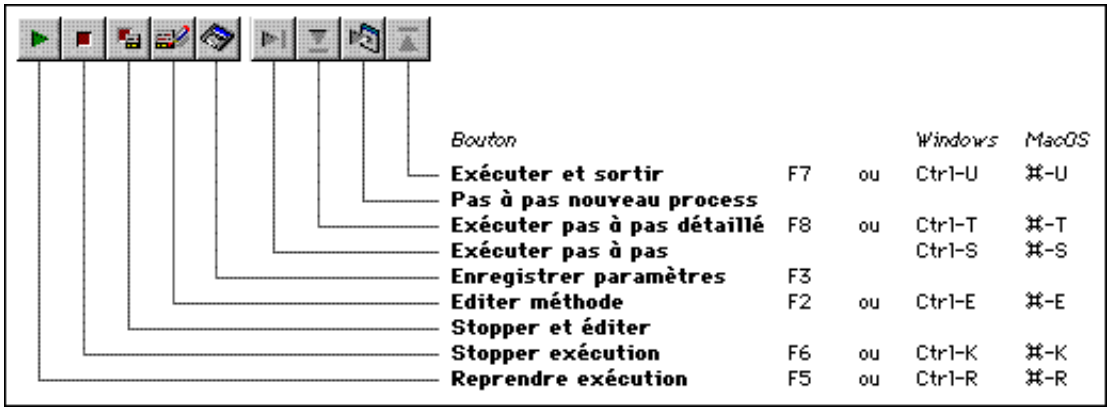
• L'ajout de conditions aux points d'arrêt sur commande ralentit l'exécution du code car la condition est évaluée à chaque fois que le point d'arrêt est rencontré. D'un autre côté, le débogage est plus rapide car 4D ignore automatiquement les occurences qui ne satisfont pas aux conditions.

• La désactivation d'un point d'arrêt sur commande produit quasiment le même effet que sa suppression. Lors de l'exécution du code, le débogueur ne passe presque pas de temps sur le point d'arrêt. L'avantage de la désactivation est que vous n'avez pas à redéfinir le point d'arrêt en cas de besoin.

Ce chapitre détaille les raccourcis disponibles dans la fenêtre du débogueur.

Barre d'outils de contrôle d'exécution

→ L'illustration ci-dessous présente les raccourcis des boutons situés dans la partie supérieure gauche de la fenêtre du débogueur :



→ La combinaison Maj+F5 ou Maj+clic sur le bouton **Reprendre exécution** reprend l'exécution et en plus désactive tous les appels TRACE ultérieurs dans le process courant.

Sous-fenêtre zone d'expression

→ un clic avec le bouton droit de la souris (Windows) ou Control+clic (Macintosh) dans la sous-fenêtre d'expression déroule le menu contextuel.

→ un double-clic sur un article de la fenêtre d'expression copie cet article dans la fenêtre d'évaluation.

Sous-fenêtre de chaîne d'appel

→ Un double-clic sur le nom d'une méthode dans la Fenêtre de chaîne d'appel affiche la méthode dans la sous-fenêtre d'évaluation des méthodes, à la ligne correspondant à la chaîne d'appel.

Sous-fenêtre d’évaluation

→ Un clic avec le bouton droit de la souris (Windows) ou Control+Clic (Macintosh) dans la fenêtre d’évaluation déroule le menu contextuel de la fenêtre.

→ Un double-clic dans la sous-fenêtre d’évaluation crée une nouvelle expression.

Fenêtre d’évaluation des méthodes

→ Un clic dans la marge gauche place ou supprime un point d’arrêt.

→ Alt+Majuscule+clic (Windows) ou Option+Majuscule+clic (Macintosh) pose un point d’arrêt provisoire

→ Alt+clic (Windows) ou Option+clic (Macintosh) affiche la fenêtre des propriétés du point d’arrêt pour un point d’arrêt nouveau ou existant.

→ Une expression ou un objet sélectionné(e) peut être copié dans la zone d’évaluation par glisser-déposer.

→ Un clic sur du texte sélectionné tout en maintenant enfoncée la touche Ctrl (Windows) ou Commande (MacOS) le copie dans la zone d’évaluation.

→ Ctrl+D (Windows) ou Commande+D (MacOS) sur un texte sélectionné le copie dans la zone d’évaluation.

Toutes les sous-fenêtres

→ Lorsqu'aucun objet n’est sélectionné dans les fenêtres, en appuyant sur **Entrée** vous avancez d’une ligne.

→ Lorsque la valeur d’un élément est sélectionnée, utilisez les touches directionnelles pour naviguer dans la liste.

→ Lorsque vous êtes en train d’éditer un élément, utilisez les touches directionnelles pour déplacer le curseur, utilisez Ctrl+A/X/C/V (Windows) ou Commande+A/X/C/V (Macintosh) en raccourci des commandes du menu **Edition** : Tout Sélectionner/Couper/ Copier/Coller.

Référence

Débogueur, Fenêtre d'évaluation, Fenêtre d'évaluation des méthodes, Fenêtre d'expression, Fenêtre de chaîne d'appel.

11

Définition structure

Les commandes de ce thème retournent la description de la structure de la base. Elles retournent le nombre de tables, le nombre de champs dans chaque table, les noms des tables et des champs, ainsi que le type et les propriétés de chaque champ.

L'identification précise de la structure de la base est très utile quand vous développez et utilisez des groupes de méthodes projets et formulaires qui peuvent être copiées dans différentes bases.

La possibilité de lire la structure de la base vous permet de développer et d'utiliser du code portable.

Référence

Champ, FIXER INDEX, LIRE PROPRIETES CHAMP, Nom de la table, Nombre de champs, Nombre de tables, Pointeurs, Table.

Nombre de tables

Définition structure

version 5

Nombre de tables → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Nombre de tables dans la base

Description

Nombre de tables retourne le nombre de tables présentes dans la base. Les tables sont numérotées dans l'ordre dans lequel elles sont créées.

Exemples

L'exemple suivant initialise les éléments du tableau tabTables. Ce tableau peut être utilisé comme une liste déroulante (ou des onglets, un zone de défilement, etc.) pour afficher dans un formulaire la liste des tables dans la base :

```
⇒   TABLEAU ALPHA (31; tabTables; Nombre de tables)
    Boucle ($vlTables; 1; Taille tableau(tabTables))
       tabTables {$vlTables} := Nom de la table ($vlTables)
    Fin de boucle
```

Référence

Nom de la table, Nombre de champs, Présentation des tableaux.

Nombre de champs

Définition structure

version 5

Nombre de champs (tableNum | tablePtr) → Numérique

Paramètre	Type		Description
tableNum | tablePtr	Num | Pointeur	→	Numéro de table ou Pointeur vers une table
Résultat	Numérique	←	Nombre de champs dans la table

Description

La commande Nombre de champs retourne le nombre de champs que contient la table dont le numéro ou le pointeur est passé dans le paramètre tableNum ou tablePtr.

Les champs sont numérotés dans l'ordre où ils ont été créés.

Exemple

La méthode projet suivante crée le tableau asChamps avec les noms des champs de la table dont le pointeur est reçu comme premier paramètre :

```
      $vlTable:=Table($1)
⇒     TABLEAU ALPHA(31;asChamps;Nombre de champs($vlTable))
      Boucle ($vlChamp;1;Taille tableau(asChamps))
         asChamps{$vlTable}:=Nom du champ($vlTable;$vlChamp)
      Fin de boucle
```

Référence

LIRE PROPRIETES CHAMP, Nom du champ, Nombre de tables, Présentation des tableaux.

Nom de la table

Définition structure

version 5

Nom de la table (numTable | ptrTable) → Alpha

Paramètre	Type		Description
numTable | ptrTable	Num | Pointeur	→	Numéro de table ou pointeur de table
Résultat	Alpha	←	Nom de la table

Description

Nom de la table retourne le nom de la table dont le numéro ou le pointeur a été passé dans numTable ou ptrTable.

Exemple

La méthode suivante est un exemple de méthode générique qui affiche les enregistrements d'une table. La référence à la table est passée en tant que pointeur vers la table. La commande Nom de la table est utilisée pour inclure le nom de la table dans la barre de titre de la fenêtre :

```
` METHODE PROJET AFFICHER SELECTION COURANTE
` AFFICHER SELECTION COURANTE (Pointeur)
` AFFICHER SELECTION COURANTE (->[Table] )

   ` Définir le titre de la fenêtre
⇒  CHANGER TITRE FENETRE(" Enregistrements pour "+Nom de la table($1))
   VISUALISER SELECTION($1>>)  ` Afficher la sélection
```

Référence

Nom du champ, Nombre de tables, Table.

Nom du champ (champPtr | tableNum {; champNum}) → Alpha

Paramètre	Type		Description
champPtr | tableNum	Numérique	→	Pointeur vers un champ ou Numéro de table
champNum	Numérique	→	Numéro de champ si un numéro de table est passé en premier paramètre
Résultat	Alpha	←	Nom du champ

Description

La commande Nom du champ retourne le nom du champ dont vous avez passé le pointeur dans champPtr, ou dont vous avez passé les numéros de table et de champ dans tableNum et champNum.

Exemples

(1) L'exemple suivant assigne au second élément du tableau ChampTableau{1} (ChampTableau étant un tableau à deux dimensions) le nom du second champ de la première table :

```
⇒ ChampTableau{1}{2}:=Nom du champ(1;2)
```

(2) L'exemple suivant assigne au second élément du tableau ChampTableau{1} (ChampTableau étant un tableau à deux dimensions) le nom du champ [MaTable]MonChamp :

```
⇒ ChampTableau{1}{2}:=Nom du champ(->[MaTable]MonChamp)
```

(3) L'exemple suivant affiche une boîte de dialogue d'alerte. Nous passons à cette méthode un pointeur vers un champ :

```
⇒ ALERTE("Le numéro du champ "+Nom du champ($1)+" de la table "
                +Nom de la table(Table($1))+" doit faire plus de cinq caractères.")
```

Référence

Champ, Nom de la table, Nombre de champs.

Table (numTable | unPtr) → Num | Pointeur

Paramètre	Type		Description
numTable | unPtr	Num | Pointeur	→	Numéro de table ou Pointeur de table ou Pointeur de champ
Résultat	Num | Pointeur	←	Pointeur de table si un N° de table est passé, N° de table si un Pointeur de table est passé, N° de table si un Pointeur de champ est passé

Description

Table a trois syntaxes différentes.
• Si vous passez un n° de table dans numTable, Table retourne un pointeur sur la table.
• Si vous passez un pointeur de table dans unPtr, Table retourne le n° de la table.
• Si vous passez un pointeur de champ dans unPtr, Table retourne le n° de table du champ.

Exemples

(1) Dans cet exemple, la variable ptrTable reçoit un pointeur sur la table n°3 :

```
⇒ ptrTable := Table (3)
```

(2) Si vous passez ptrTable à la fonction Table, elle retourne 3. Par exemple, dans la ligne suivante, la variable numTable prend la valeur 3 :

```
⇒ numTable := Table (ptrTable)
```

(3) Dans l'exemple suivant, la variable numTable est égale au n° de la table [Table3] :

```
⇒ numTable := Table (->[Table3])
```

(4) Dans l'exemple suivant, la variable numTable est égale au numéro de la table à laquelle appartient le champ [Table3]Champ1 :

```
⇒ numTable := Table (->[Table3]Champ1)
```

Référence

Champ, Nom de la table, Nombre de tables, Pointeurs.

LIRE PROPRIETES TABLE (ptrTable|numTable; invisible{; trigSvgdeNouv{; trigSvgdeEnr{; trigSupprEnr{; trigChargEnr}}}})

Paramètre	Type		Description
ptrTable | numTable	Pointeur | Entier long	→	Pointeur de table ou Numéro de table
invisible	Booléen	←	Vrai = Invisible, Faux = Visible
trigSvgdeNouv	Booléen	←	Vrai = Trigger “Sur sauvegarde nouvel enreg” activé, sinon Faux
trigSvgdeEnr	Booléen	←	Vrai = Trigger “Sur sauvegarde enregistrement” activé, sinon Faux
trigSupprEnr	Booléen	←	Vrai = Trigger “Sur suppression enreg” activé, sinon Faux
trigChargEnr	Booléen	←	Vrai = Trigger “Sur chargement enregistrement” activé, sinon Faux

Description

La commande LIRE PROPRIETES TABLE retourne les propriétés de la table désignée par ptrTable ou numTable. Vous pouvez passer dans le premier paramètre soit un pointeur vers la table, soit le numéro de la table.

Après l’exécution de la commande :
• Le paramètre invisible retourne Vrai si la table dispose de l’attribut “Invisible”, Faux sinon. L’attribut “Invisible” permet de masquer la table dans les éditeurs standard de 4D (étiquettes, graphes...).
• Le paramètre trigSvgdeNouv retourne Vrai si le trigger “Sur sauvegarde nouvel enreg” a été activé pour la table, Faux sinon.
• Le paramètre trigSvgdeEnr retourne Vrai si le trigger “Sur sauvegarde enregistrement” a été activé pour la table, Faux sinon.
• Le paramètre trigSupprEnr retourne Vrai si le trigger “Sur suppression enreg” a été activé pour la table, Faux sinon.
• Le paramètre trigChargEnr retourne Vrai si le trigger “Sur chargement enregistrement” a été activé pour la table, Faux sinon.

Référence

LIRE PROPRIETES CHAMP, LIRE PROPRIETES LIEN, LIRE PROPRIETES SAISIE CHAMP.

Champ (tableNum | champPtr{; champNum}) → Num | Pointeur

Paramètre	Type		Description
tableNum | champPtr	Num | Pointeur	→	Numéro de table ou Pointeur de champ
champNum	Numérique	→	Numéro de champ si un numéro de table est passé
Résultat	Num | Pointeur	←	Numéro de champ si un pointeur de champ est passé, Pointeur de champ si des numéros de table et de champ sont passés

Description

La commande Champ a deux syntaxes :
• Si vous passez un numéro de table dans tableNum et un numéro de champ dans champNum, Champ retourne un pointeur vers le champ.
• Si vous passez un pointeur vers un champ dans champPtr, Champ retourne le numéro du champ.

Exemples

(1) L'exemple suivant assigne la variable champPtr à un pointeur vers le deuxième champ de la troisième table :

```
⇒ champPtr:=Champ(3; 2)
```

(2) Si vous passez champPtr (un pointeur vers le 2e champ de la table) à Champ, la valeur 2 est retournée. La ligne suivante assigne la valeur 2 à champNum :

```
⇒ champNum:=Champ(champPtr)
```

(3) Dans cet exemple, la variable champNum est égale au numéro de champ de [Table3]Champ2 :

```
⇒ champNum:=Champ(->[Table3]Champ2)
```

Référence

LIRE PROPRIETES CHAMP, Nom du champ, Nombre de champs, Table.

LIRE PROPRIETES CHAMP (chpPtr | tableNum{; champNum}; champType{; champLong{; indexé}})

Paramètre	Type		Description
chpPtr | tableNum	Pointeur | Num	→	Pointeur de champ ou Numéro de table
champNum	Numérique	→	Numéro de champ si un numéro de table est passé en premier caractère
champType	Numérique	←	Type de champ
champLong	Numérique	←	Longueur du champ (si alphanumérique)
indexé	Booléen	←	VRAI = Indexé, FAUX = Non indexé

Description

La commande LIRE PROPRIETES CHAMP retourne des informations sur le champ désigné par tableNum et champNum ou par chpPtr.

Vous pouvez soit passer :
• les numéros de table et de champ dans tableNum et champNum
• ou un pointeur vers le champ dans chpPtr.

Après l'appel :

• Le paramètre champType retourne le type du champ. Le paramètre variable champType reçoit l'une des valeurs prédéfinies par les constantes de 4e Dimension :

Constante	Type	Valeur
Est un champ alpha	Entier long	0
Est un texte	Entier long	2
Est un numérique	Entier long	1
Est un entier	Entier long	8
Est un entier long	Entier long	9
Est une date	Entier long	4
Est une heure	Entier long	11
Est un booléen	Entier long	6
Est une image	Entier long	3
Est une sous table	Entier long	7
Est un BLOB	Entier long	30

• Le paramètre champLong retourne la longueur du champ si celui-ci est de type Alpha (ce qui signifie que vous obtenez champType=Est un champ alpha). La valeur de champLong n'est pas significative pour les autres types de champ.

• Le paramètre indexé retourne Vrai si le champ est indexé, Faux sinon. La valeur de indexé est significative pour les champs de type Alphanumérique, Entier, Entier long, Réel, Date, Heure et Booléen.

Exemples

(1) Dans l'exemple suivant, les variables vType, vLong et vIndex prennent pour valeur les propriétés du troisième champ de la première table :

⇒ **LIRE PROPRIETES CHAMP**(1; 3;vType;vLong;vIndex)

(2) L'exemple suivant récupère dans les variables vType, vLong et vIndex les propriétés du champ [Table3]Champ2 :

⇒ **LIRE PROPRIETES CHAMP**(->[Table3]Champ2;vType;vLong;vIndex)

Référence

Champ, FIXER INDEX, Nom du champ.

LIRE PROPRIETES SAISIE CHAMP

Définition structure

version 6.7

LIRE PROPRIETES SAISIE CHAMP (ptrChp | numTable{; numChamp}; nomEnum{; obligatoire{; nonSaisissable{; nonModifiable}})

Paramètre	Type		Description
ptrChp | numTable	Pointeur | Entier long	→	Pointeur de champ ou Numéro de table
numChamp	Entier long	→	Numéro de champ si un numéro de table est passé en premier paramètre
nomEnum	Alpha	←	Nom de l'énumération associée ou Chaîne vide
obligatoire	Booléen	←	Vrai = Obligatoire, Faux = Facultatif
nonSaisissable	Booléen	←	Vrai = Non saisissable, Faux = Saisissable
nonModifiable	Booléen	←	Vrai = Non modifiable, Faux = Modifiable

Description

La commande LIRE PROPRIETES SAISIE CHAMP retourne les propriétés relatives à la saisie de données du champ désigné par numTable et numChamp ou par ptrChp.
Vous pouvez passer :
• soit des numéros de table et de champ dans numTable et numChamp,
• soit un pointeur vers le champ dans ptrChp.

Les propriétés retournées par cette commande sont celles qui ont été définies au niveau de la fenêtre de structure de la base. Des propriétés similaires peuvent également être définies au niveau des formulaires.

Après l'exécution de la commande :
• Le paramètre nomEnum contient le nom de l'énumération associée au champ, s'il y en a une. Il est possible d'associer un énumération aux champs de type Alpha, Texte, Numérique, Entier, Entier long, Date, Heure et Booléen. Si aucune énumération n'est associée au champ, ou si son type n'admet pas l'association d'énumération, une chaîne vide ("") est retournée.
• Le paramètre obligatoire retourne Vrai si le champ dispose de l'attribut "Obligatoire", Faux sinon.
• Le paramètre nonSaisissable retourne Vrai si le champ dispose de l'attribut "Non saisissable", Faux sinon. Un champ non saisissable ne peut qu'être lu, il n'accepte aucune saisie de données.
• Le paramètre nonModifiable retourne Vrai si le champ dispose de l'attribut "Non modifiable", Faux sinon. Un champ non modifiable n'accepte qu'une seule saisie, et ne peut plus être modifié par la suite.
Les attributs "Obligatoire", "Non saisissable" et "Non modifiable" peuvent être associés aux champs de tous types, hormis Sous-table et BLOB.

Référence

LIRE PROPRIETES CHAMP, LIRE PROPRIETES LIEN, LIRE PROPRIETES TABLE.

LIRE PROPRIETES LIEN (ptrChp | numTable{; numChamp}; tableDest; champDest{; discriminant{; allerAuto{; retourAuto}})

Paramètre	Type		Description
ptrChp | numTable	Pointeur | Entier long	→	Pointeur de champ ou Numéro de table
numChamp	Entier long	→	Numéro de champ si un numéro de table est passé en premier paramètre
tableDest	Entier long	←	Numéro de la table cible ou 0 si aucun lien ne part du champ
champDest	Entier long	←	Numéro du champ cible ou 0 si aucun lien ne part du champ
discriminant	Entier long	←	Numéro du champ discriminant ou 0 si aucun champ discriminant
allerAuto	Booléen	←	Vrai = Lien aller automatique, Faux = Lien aller manuel
retourAuto	Booléen	←	Vrai = Lien retour automatique, Faux = Lien retour manuel

Description

La commande LIRE PROPRIETES LIEN retourne les propriétés du lien, s'il y en a un, qui part du champ source, désigné par numTable et numChamp ou par ptrChp.
Vous pouvez passer :
• soit des numéros de table et de champ dans numTable et numChamp,
• soit un pointeur vers le champ dans ptrChp.

Après l'exécution de la commande :
• Les paramètres tableDest et champDest contiennent respectivement le numéro de la table et du champ vers lesquels pointe le lien partant du champ source. Si aucun lien ne part du champ, ces paramètres contiennent 0.
• Le paramètre discriminant contient le numéro du champ discriminant (appartenant à la table cible) défini pour le lien. Si aucun champ discriminant n'a été défini pour le lien ou si aucun lien ne part du champ source, ce paramètre contient 0.
• Les paramètres allerAuto et retourAuto retournent Vrai si respectivement les options "Lien aller auto" et "Lien retour auto" ont été cochées pour le lien, Faux sinon.

Note : Les deux derniers paramètres retournent également Vrai si aucun lien ne part du champ source (dans ce cas, leur valeur est non significative). La valeur des paramètres tableDest et champDest vous permet de vous assurer de l'existence d'un lien.

Référence

LIRE PROPRIETES CHAMP, LIRE PROPRIETES SAISIE CHAMP, LIRE PROPRIETES TABLE.

FIXER INDEX (champ; index{; mode}{; *})

Paramètre	Type		Description
champ	Champ | sous-chp	→	Champ duquel créer ou supprimer l'index
index	Booléen	→	Créer l'index (Vrai) ou supprimer l'index (Faux)
mode	Entier long	→	Utiliser mode d'indexation rapide (pourcentage)
*		→	Indexation asynchrone si * est passé

Description

La commande FIXER INDEX crée ou supprime l'index du champ ou sous-champ que vous avez passé dans champ.

Pour indexer le champ ou le sous-champ, passez Vrai dans index. Si l'index existe déjà, la commande ne fait rien. Pour supprimer l'index, passez Faux dans index. S'il n'existe pas d'index, la commande ne fait rien.

FIXER INDEX n'indexera pas les enregistrements verrouillés ; la commande attendra que les enregistrements soient libérés.

A compter de la version 6.5 de 4D, deux modes d'indexation sont utilisables : le **mode rapide** et le **mode classique** (pour plus d'informations sur les modes d'indexation proposés par 4D, reportez-vous au manuel *Mode Structure*). Vous pouvez choisir le mode d'indexation à utiliser en passant ou non le paramètre mode. Ce paramètre n'est appelé que si la commande entraîne effectivement la création de l'index (c'est-à-dire si le paramètre index est Vrai).

• Si vous ne passez pas le paramètre mode, l'indexation sera effectuée en **mode classique**. Dans ce cas, comme l'indexation est effectuée dans un process séparé, la base de données reste utilisable pendant l'opération. Si une action utilisant l'index est exécutée alors que l'index est en train d'être construit, il ne sera pas utilisé. Pour savoir si un champ est indexé, utilisez la commande LIRE PROPRIETES CHAMP.

• Si vous passez le paramètre mode, la commande utilisera le **mode rapide**. Dans ce cas, il n'est pas possible de modifier les données de la table pendant la durée de l'indexation. Vous devez passer dans mode une valeur entière représentant un pourcentage. Cette valeur vous permet d'indiquer le type d'usage pour lequel vous souhaitez que l'index soit le plus performant. Elle doit être comprise dans l'intervalle 0 < mode < 100.
- Plus mode est proche de 0, plus l'index sera performant lors des ajouts ou des insertions d'enregistrements.
- Plus mode est proche de 100, plus l'index sera performant lors des recherches.

Le paramètre optionnel * indique une indexation asynchrone (simultanée). Une indexation asynchrone permet à la méthode appelante de poursuivre son exécution immédiatement après l'appel, que l'indexation soit terminée ou non. Cependant, l'exécution sera stoppée si une commande requiert l'index.

Exemples

(1) L'exemple suivant indexe le champ [Clients]Num avec le mode classique :

```
     LIBERER ENREGISTREMENT([Clients])
⇒    FIXER INDEX ([Clients]Num; Vrai)
```

(2) Vous souhaitez indexer le champ [Clients]Nom, en mode rapide. Ce champ est principalement utilisé pour effectuer des recherches :

```
⇒    FIXER INDEX([Clients]Nom;Vrai;100)
```

(3) Vous souhaitez indexer le champ [Prospects]Nom, en mode rapide. Le champ sera utilisé en ajout et en insertion, mais également pour des recherches. Toutefois, le premier type d'utilisation restera le plus fréquent :

```
⇒    FIXER INDEX([Prospects]Nom;Vrai;30)
```

Référence

CHERCHER, LIRE PROPRIETES CHAMP, TRIER.

Lire parametre base

Définition structure

version 6.7 (Modifiée)

Lire parametre base ({table; }sélecteur) → Entier long

Paramètre	Type		Description
table	Table	→	Table du paramètre ou Table par défaut si ce paramètre est omis
sélecteur	Entier long	→	Code du paramètre de la base
Résultat	Entier long	←	Valeur du paramètre

Description

La commande Lire parametre base permet de lire la valeur courante des paramètres de la base 4D, pour le process courant.

sélecteur désigne le paramètre de la base à lire. 4e Dimension vous propose les constantes prédéfinies suivantes, placées dans le thème “Paramètres de la base” :

Constante	Type	Valeur
Ratio de tri séq	Entier long	1
Optimisation accès séq	Entier long	2
Ratio valeurs distinctes séq	Entier long	3
Compression index	Entier long	4
Ratio chercher dans sélec séq	Entier long	5
Minimum process Web	Entier long	6
Maximum process Web	Entier long	7
Mode conversion Web	Entier long	8
Taille cache données	Entier long	9
Appels système 4e Dimension	Entier long	10
Appels système 4D Server	Entier long	11
Appels système 4D Client	Entier long	12
Timeout 4D Server	Entier long	13
Timeout 4D Client	Entier long	14
Numéro du port	Entier long	15
Adresse IP d’écoute	Entier long	16
Jeu de caractères	Entier long	17
Process Web simultanés maxi	Entier long	18

Les valeurs pouvant être retournées par cette fonction pour les sélecteurs 1 à 8 et 10 à 18 sont détaillées dans la description de la commande FIXER PARAMETRE BASE.

Le sélecteur Taille cache données (9) vous permet d'obtenir la taille courante du cache mémoire utilisé par 4D pour les données. Cette valeur est exprimée en octets.
La taille du cache est issue des paramétrages définis dans la page "Réglages système" des Propriétés de la base (paramètres **Cache maximum** ainsi que, sous MacOS uniquement, **Cache minimum**). La taille réelle allouée au cache dépend de ces paramètres mais également de l'état des ressources mémoire de la machine.
Ce sélecteur vous permet donc de connaître précisément la taille courante de la mémoire allouée au cache par 4D.

Note : La taille du cache de données ne peut pas être fixée par programmation. Autrement dit, il n'est pas possible d'utiliser le sélecteur Taille cache données avec la commande FIXER PARAMETRE BASE.

Exemples

(1) Cette méthode permet de récupérer les valeurs courantes du minuteur interne de 4D :

```
C_ENTIER LONG($ticksbtwcalls;$maxticks;$minticks;$lparams)
Si (Type application=4e Dimension) ` Si nous sommes en 4D monoposte
⇒    $lparams:=Lire parametre base(Appels système 4e Dimension)
     $ticksbtwcalls:=$lparams & 0x00ff
     $maxticks:=($lparams>>8) & 0x00ff
     $minticks:=($lparams>>16) & 0x00ff
Fin de si
```

(2) Le sélecteur 16 (Adresse IP d'écoute) permet d'obtenir l'adresse IP sur laquelle le serveur Web 4D reçoit les requêtes HTTP. L'adresse obtenue est de forme hexadécimale. L'exemple suivant permet de décomposer l'adresse IP reçue :

```
C_ENTIER LONG($a;$b;$c;$d)
C_ENTIER LONG($addr)
⇒ $addr:=Lire parametre base(Adresse IP d'écoute)
$a:=($addr>>24)&0x000000ff
$b:=($addr>>16)&0x000000ff
$c:=($addr>>8)&0x000000ff
$d:=$addr&0x000000ff
```

Référence

CHERCHER DANS SELECTION, FIXER PARAMETRE BASE, VALEURS DISTINCTES.

FIXER PARAMETRE BASE

Définition structure

version 6.7 (Modifiée)

FIXER PARAMETRE BASE ({table; }sélecteur; valeur)

Paramètre	Type		Description
table	Table	→	Table à paramétrer ou Table par défaut si ce paramètre est omis
sélecteur	Entier long	→	Code du paramètre de la base à modifier
valeur	Entier long	→	Valeur du paramètre

Description

La commande FIXER PARAMETRE BASE permet de modifier divers paramètres internes de la base de données 4D, pour le process courant.

sélecteur désigne le paramètre à modifier. 4e Dimension vous propose les constantes prédéfinies suivantes, placées dans le thème “Paramètres de la base” :

Constante	Type	Valeur
Ratio de tri séq	Entier long	1
Optimisation accès séq	Entier long	2
Ratio valeurs distinctes séq	Entier long	3
Compression index	Entier long	4
Ratio chercher dans sélec séq	Entier long	5
Minimum process Web	Entier long	6
Maximum process Web	Entier long	7
Mode conversion Web	Entier long	8
Appels système 4e Dimension	Entier long	10
Appels système 4D Server	Entier long	11
Appels système 4D Client	Entier long	12
Timeout 4D Server	Entier long	13
Timeout 4D Client	Entier long	14
Numéro du port	Entier long	15
Adresse IP d’écoute	Entier long	16
Jeu de caractères	Entier long	17
Process Web simultanés maxi	Entier long	18

valeur désigne la valeur du paramètre. La valeur à passer dépend du paramètre que l’on souhaite modifier. Voici les valeurs possibles pour chaque sélecteur :

Sélecteur = 1 (Ratio de tri séq)
• Valeurs possibles : 0 -> 100 000
• Description : Ratio d’enregistrements sélectionnés (par rapport au nombre total d’enregistrements) au-dessous duquel les tris sont effectués en mode séquentiel. Ce ratio doit être exprimé sur 100 000. La valeur par défaut est 9 000 (soit 9 %).

Sélecteur = 2 (Optimisation accès séq)
• Valeurs possibles : 0 ou 1 (0 = non optimisé, 1 = optimisé)
• Description : Mode d'optimisation pour les accès séquentiels (tris, recherches, SELECTION VERS TABLEAU). Par défaut, la valeur est 1 (mode optimisé). En mode optimisé, 4D tente de lire depuis le disque plusieurs enregistrements en une fois. Ce mode est particulièrement intéressant lorsque la taille du cache est faible.

Sélecteur = 3 (Ratio valeurs distinctes séq)
• Valeurs possibles : 0 -> 100 000
• Description : Ratio d'enregistrements sélectionnés (par rapport au nombre total d'enregistrements) au-dessous duquel la commande VALEURS DISTINCTES sera exécutée en mode séquentiel. Ce ratio doit être exprimé sur 100 000.

Sélecteur = 4 (Compression index)
• Valeurs possibles : 0 ou 1 (0 = non, 1 = oui)
• Description : Activation ou non de la compression des pages d'index. Par défaut, la valeur est 1 (les index sont compressés si nécessaire).
Les pages d'index peuvent, dans des bases contenant beaucoup d'index et d'enregistrements, consommer beaucoup de place dans la mémoire cache de 4D. Lorsque le cache est plein et que 4D a besoin de place supplémentaire, les données se trouvant dans le cache ne sont plus directement déchargées : avant d'effectuer cette opération, le programme va contrôler s'il peut gagner de la place en compactant des pages d'index. Cette alternative permet d'éviter le rechargement ultérieur des données.

Sélecteur = 5 (Ratio chercher dans sélec séq)
• Valeurs possibles : 0 -> 100 000
• Description : Ratio d'enregistrements sélectionnés (par rapport au nombre total d'enregistrements) au-dessous duquel la commande CHERCHER DANS SELECTION sera exécutée en mode séquentiel. Ce ratio doit être exprimé sur 100 000.

Sélecteur = 6 (Minimum process Web)
• Valeurs possibles : 0 -> 32 767
• Description : Nombre minimum de process Web à maintenir en mode sans contexte. Par défaut, la valeur est 0 (cf. ci-dessous).

Sélecteur = 7 (Maximum process Web)
• Valeurs possibles : 0 -> 32 767
• Description : Nombre maximum de process Web à maintenir en mode sans contexte. Par défaut, la valeur est 10.
Afin que le serveur Web soit réactif, en mode sans contexte, 4D maintient endormis les process Web pendant 5 secondes, et les réutilise pour traiter les éventuelles requêtes HTTP suivantes. En termes de performances, ce principe est en effet plus avantageux que la création d'un nouveau process à chaque requête. Une fois un process Web réutilisé, il est à nouveau endormi pour 5 secondes, sauf si le nombre maximum de process Web est dépassé (auquel cas il est tué). Si aucune requête n'a été attribuée à un process Web durant les 5 secondes, il est tué, sauf si un nombre minimum de process Web a été fixé et est atteint (auquel cas il est à nouveau endormi).
Ces paramètres vous permettent d'ajuster le fonctionnement de votre serveur Web en fonction du nombre de requêtes, de la mémoire disponible, etc.

Sélecteur = 8 (Mode conversion Web)
• Valeurs possibles : 0, 1, 2 ou 3
0 (mode par défaut) = Conversion au format HTML 4.0 si le browser le permet. Sinon, format HTML 3.2 + usage de tableaux.
1 = Conversion au format HTML 2.0 (mode 6.0.x),
2 = Conversion au format HTML 3.2 (mode 6.5),
3 = Conversion au format HTML 4.0 + CSS-P (depuis version 6.5.2).
• Description : Mode de conversion des formulaires 4D pour le Web.
Par défaut, le serveur Web 4D 6.7 utilise les feuilles de style en cascade (CSS1) pour générer des pages HTML dont l'apparence est très proche de celle des formulaires obtenus dans 4e Dimension.
Dans le cas des bases créées avec des versions antérieures de 4D, ce fonctionnement peut perturber la conversion correcte des formulaires. Vous pouvez donc modifier le mode de conversion des formulaires.
Ce mode n'est utilisé que pour le process (contexte Web) dans lequel la commande FIXER PARAMETRE BASE est appelée. Elle peut être placée dans la Méthode base Sur connexion Web pour assurer la conversion des formulaires de la base, ou juste avant l'appel d'un formulaire particulier. Hors du mode contextuel ou d'un process Web, la commande est sans effet.

Note : La fonction Lire parametre base admet un sélecteur supplémentaire, Taille cache données (9). Ce paramètre ne peut pas être utilisé avec la commande FIXER PARAMETRE BASE. Pour plus d'informations, reportez-vous à la description de la fonction Lire parametre base.

Sélecteur = 10 (Appels système 4e Dimension)
Sélecteur = 11 (Appels système 4D Server)
Sélecteur = 12 (Appels système 4D Client)
• Valeurs possibles : pour ces trois sélecteurs, le paramètre valeur est de la forme hexadécimale 0x00aabbcc dans laquelle :
aa = nombre minimum de ticks par appel au système (de 0 à 100 inclus)
bb = nombre maximum de ticks par appel au système (de 0 à 100 inclus)
cc = nombre de ticks entre deux appels au système (de 0 à 20 inclus).
Si une des valeurs est hors de son intervalle, 4D la fixe à son maximum.
Vous pouvez également passer dans valeur une des trois valeurs suivantes, correspondant à un ensemble de réglages standard :
valeur = -1 : priorité maximum allouée à 4D,
valeur = -2 : priorité moyenne allouée à 4D,
valeur = -3 : priorité minimum allouée à 4D.
• Description : Ce sélecteur vous permet de fixer dynamiquement les réglages des appels système de 4D. En fonction du Sélecteur, le gestionnaire d'appels système sera défini pour :
- 4e Dimension (monoposte) lorsque la commande est appelée depuis 4e Dimension, ainsi qu'à 4D Tools (sélecteur=10).
- 4D Server lorsque la commande est appelée depuis 4D Server (sélecteur=11).
- 4D Client lorsque la commande est appelée depuis 4D Client (sélecteur=12).
Reportez-vous à l'exemple (1).

Sélecteur = 13 (Timeout 4D Server)
• Description : Ce paramètre permet de modifier la valeur du délai avant déconnexion (timeout) accordé par 4D Server aux postes clients.
Par défaut, la valeur du délai avant déconnexion utilisée par 4D Server est définie dans la page “Connexions” de la boîte de dialogue des Propriétés de la base, sur le poste serveur. Le sélecteur Timeout 4D Server vous permet de fixer, à l’aide du paramètre valeur, un nouveau timeout, exprimé en minutes. Cette possibilité permet en particulier d’augmenter la valeur du timeout avant l’exécution sur le poste client d’une opération bloquante et de longue durée, risquant d’entraîner une déconnexion ; par exemple, l’impression d’un grand nombre de pages.

Vous disposez en outre de deux possibilités :
- effectuer une modification *globale* et *permanente* : la nouvelle valeur s’applique à tous les process et est stockée dans les préférences de l’application (équivaut à une modification de la valeur dans la boîte de dialogue des Propriétés de la base). Pour cela, passez une valeur **positive** dans le paramètre valeur.
- effectuer une modification *restreinte* et *temporaire* : la nouvelle valeur ne s’applique qu’au process appelant (les autres process conservant la valeur d’origine), et est abandonnée dès que le serveur reçoit un signe d’activité du poste client — par exemple, dès que l’opération est terminée. Cette possibilité est utile pour gérer les opérations longues initiées par des plug-ins. Pour cela, passez une valeur **négative** dans le paramètre valeur.
Pour définir une connexion “Ouverte en permanence”, passez 0 dans valeur.
Reportez-vous à l’exemple (2).

Sélecteur = 14 (Timeout 4D Client)
• Description : Ce paramètre permet de modifier la valeur du délai avant déconnexion (timeout) accordé par le poste 4D Client au poste 4D Server.
Par défaut, la valeur du délai avant déconnexion utilisée par 4D Client est définie dans la page “Connexions” de la boîte de dialogue des Propriétés de la base, sur le poste client.
Pour plus d’informations sur le fonctionnement de ce sélecteur, reportez-vous ci-dessus à la description du sélecteur Timeout 4D Server (13).
Le sélecteur Timeout 4D Client est à utiliser dans des cas très spécifiques.

Sélecteur = 15 (Numéro du port)
• Description : Ce paramètre permet de modifier par programmation le numéro du port TCP utilisé par le serveur Web 4D. Par défaut, la valeur est 80.
Le numéro de port TCP est défini dans la page “Serveur Web I” de la boîte de dialogue des Propriétés de la base.
Le sélecteur Numéro du port est utile dans le cadre de serveurs Web 4D compilés et fusionnés avec 4D Engine (pas d’accès au mode Structure). Pour plus d’informations sur le numéro de port TCP, reportez-vous à la section Services Web, Configuration.

Sélecteur = 16 (Adresse IP d’écoute)
• Description : Ce paramètre permet d’indiquer par programmation l’adresse IP sur laquelle le serveur Web doit recevoir les requêtes HTTP. Par défaut, aucune adresse particulière n’est spécifiée (valeur = 0).
Ce paramètre est défini dans la page “Serveur Web I” de la boîte de dialogue des Propriétés de la base.

Le sélecteur Adresse IP d'écoute est utile dans le cadre de serveurs Web 4D compilés et fusionnés avec 4D Engine (pas d'accès au mode Structure).
Passez dans le paramètre valeur l'adresse IP sous forme hexadécimale. Ainsi, pour désigner une adresse du type "a.b.c.d", le code sera de la forme :

```
C_ENTIER LONG($addr)
$addr:=($a<<24)|($b<<16)|($c<<8)|$d
⇒ FIXER PARAMETRE BASE(Adresse IP d'écoute;$addr)
```

Reportez-vous également à l'exemple (3). Pour plus d'informations sur la désignation de l'adresse IP, reportez-vous à la section Services Web, Paramétrages du Serveur Web.

Sélecteur = 17 (Jeu de caractères)
• Valeurs :
0 : Occidental
1 : Japonais
2 : Chinois
3 : Coréen
4 : Défini par l'utilisateur
5 : *Réservé*
6 : Europe Centrale
7 : Cyrillique
8 : Arabe
9 : Grec
10 : Hébreu
11 : Turc
12 : Nordique
• Description : Ce paramètre permet d'indiquer par programmation le jeu de caractères que le serveur Web 4D doit utiliser pour communiquer avec les navigateurs Web qui se connectent à la base. La valeur par défaut dépend de la langue du système d'exploitation. Ce paramètre est défini dans la page "Serveur Web II" de la boîte de dialogue des Propriétés de la base. Le sélecteur Jeu de caractères est utile dans le cadre de serveurs Web 4D compilés et fusionnés avec 4D Engine (pas d'accès au mode Structure).

Sélecteur = 18 (Process Web simultanés maxi)
• Valeurs : Vous pouvez passer toute valeur incluse entre 10 et 32 000. La valeur par défaut est 32 000.
• Description : Ce paramètre permet permet de définir la limite strictement supérieure du nombre de process Web de tout type (contextuels, non contextuels ou appartenant à la réserve de process — cf. à ce sujet le sélecteur 7, Maximum process Web) acceptés par le serveur Web. Lorsque ce nombre limite (moins un) est atteint, 4D ne crée plus de nouveau process et retourne le message "Serveur non disponible" (statut HTTP 503 - Service Unavailable) à toute nouvelle requête.
Ce paramètre permet de prévenir la saturation du serveur Web 4D pouvant se produire lors d'un envoi massif de requêtes ou d'une demande excessive de création de contextes. Il peut également être défini dans la boîte de dialogue des Propriétés de la base (cf. section Serveur Web, Paramétrages du serveur Web).

En théorie, le nombre maximum de process Web est le résultat de la division Mémoire disponible / Taille de la pile d'un process Web. Une autre solution consiste à visualiser les informations sur les process Web affichées dans l'Exporateur d'exécution : le nombre courant de process Web et le nombre maximum atteint depuis le démarrage du serveur Web sont indiqués.

Note : Si vous passez une valeur inférieure à la limite supérieure de la "réserve" de process (sélecteur 7), cette limite est réduite à la valeur du sélecteur 18. Si nécessaire, la limite inférieure de la réserve (sélecteur 6) est également ajustée.

Exemples

(1) Cette méthode permet de définir les valeurs du minuteur des **appels système** si l'application courante est 4e Dimension monoposte :

```
C_ENTIER LONG($ticksbtwcalls;$maxticks;$minticks;$lparams)
Si (Type application=4e Dimension) ` Si nous sommes en 4D monoposte
   $ticksbtwcalls:=12
   $maxticks:=20
   $minticks:=7
   $lparams:=($minticks<<16)|($maxticks<<8)|$ticksbtwcalls
⇒  FIXER PARAMETRE BASE(Appels système 4e Dimension;$lparams)
Fin de si
```

(2) L'instruction suivante permet d'anticiper un éventuel problème de **timeout** :

```
   `Augmentation du timeout à 3 heures pour le process courant
⇒  FIXER PARAMETRE BASE(Timeout 4D Server;-60*3)
   `Exécution d'une opération longue hors du contrôle de 4D
...
WR MAILING (LaZone;3;0)
...
```

(3) L'**adresse IP** 192.193.194.195 sera fixée avec l'instruction suivante :

```
⇒  FIXER PARAMETRE BASE(Adresse IP d'écoute;0xC0C1C2C3)
```

Référence

CHERCHER DANS SELECTION, Lire parametre base, VALEURS DISTINCTES.

12

Documents système

Introduction

Tous les documents et applications que vous utilisez sur votre ordinateur sont stockés en tant que **fichiers** sur le ou les disques durs **connectés** ou **montés** sur votre ordinateur, ou encore sur des disquettes et autres supports de stockage permanent. Dans cette documentation ainsi que dans 4D, les termes **fichier** ou **document** sont indifféremment employés pour désigner ces documents et applications. Cependant, la plupart des commandes de ce thème utilisent le mot document car, généralement, vous les utiliserez pour accéder à des documents (par opposition à des fichiers d'application ou des fichiers système) sur disque.

Un disque dur peut être formaté de manière à comporter une ou plusieurs partitions. Chaque partition s'appelle un **volume**. Peu importe que ces volumes soient des partitions physiquement présentes sur le même disque dur ou non, au niveau de 4D, ces volumes sont considérés comme des entités séparées et équivalentes.

Un volume peut être situé sur un disque dur physiquement connecté à votre machine ou monté par le réseau par l'intermédiaire d'un protocole de partage de fichiers tel que NetBEUI (Windows) ou AFP (Macintosh). Quel que soit le cas, au niveau de 4D, ces volumes sont considérés de la même façon lorsque vous utilisez les commandes du thème Documents Système (à moins que vous n'en décidiez autrement et utilisiez des plug-ins 4D pour étendre les capacités de votre application dans ce domaine).

Chaque volume a un **nom de volume**. Sous Windows, les volumes sont désignés par une lettre suivie de deux points. Généralement, A: et B: sont utilisés pour désigner les lecteurs de disquettes et C: désigne le volume que vous utilisez pour lancer votre système (à moins que vous n'ayez configuré votre PC différemment). Ensuite, les lettres D: à Z: sont utilisées pour les volumes supplémentaires connectés à votre PC (lecteurs CD-ROM, autres lecteurs, lecteurs réseau, etc.). Sous MacOS, les volumes ont des noms communs dont la longueur maximale est de 31 caractères (ces noms sont ceux que vous visualisez sur le bureau au niveau du Finder).

Normalement, vous classez vos documents dans des **dossiers** qui peuvent eux-mêmes contenir d'autres dossiers. Il n'est pas conseillé d'accumuler des centaines ou des milliers de fichiers au même niveau d'un volume. C'est un fouillis, qui de plus ralentit votre système. Sous Windows, un dossier est parfois encore appelé un répertoire (bien que le terme “dossier” soit de plus en plus utilisé depuis l'introduction de Windows 95/98).

Pour identifier un document de manière certaine, vous avez besoin de connaître le nom du volume, le nom du ou des dossiers(s) dans le(s)quel(s) se trouve le document, ainsi que le nom du document lui-même. Si vous concaténez tous ces noms, vous obtenez le **chemin d'accès** à ce document. Dans le nom de ce chemin, les noms de dossiers sont séparés par un caractère spécial appelé **symbole séparateur** (de répertoire). Sous Windows, ce caractère est la barre oblique inversée \, sous MacOS les deux-points :

Examinons un exemple. Vous disposez d'un document Important situé dans le dossier Mémos, lui-même situé dans le dossier Documents, lui-même situé dans le dossier EnCours.

Si, sous Windows, l'ensemble est situé sur le volume C: , le chemin d'accès au document est donc :
C:\EnCours\Documents\Mémos\Important.TXT

Si, sous MacOS, l'ensemble est situé sur le volume Interne, le chemin d'accès au document est donc :
Interne:EnCours:Documents:Mémos:Important

Notez que, sous Windows, avec cet exemple, le nom du document contient le suffixe .TXT. Nous verrons pourquoi plus bas.

Quelle que soit la plate-forme, le chemin d'accès à un document peut être exprimé sous la forme suivante : VolNom DosSep { DosNom DosSep { DosNom DosSep { ... } } } DocNom.

Tous les documents (fichiers) situés sur des volumes ont plusieurs caractéristiques généralement appelées **attributs** ou **propriétés** : par exemple le **nom** du document lui-même (jusqu'à 31 caractères sur Macintosh, jusqu'à 255 caractères sous Windows 95 ou NT 4.0).

Type de document et Créateur

Sous Windows, un document a un **type**. Sous MacOS, un document a un **type** et un **créateur** (ou "creator"). Le type d'un document indique généralement ce qu'est le document ou ce qu'il contient. Par exemple, un document de type texte contient du texte (sans style). Sous Windows, le type d'un document est déterminé par son suffixe, appelé **extension de fichier**, rattaché au nom du document. Par exemple .TXT est l'extension de fichier Windows pour des documents texte. Sous MacOS, le type d'un document est déterminé par la propriété **type de fichier** du document qui est une signature sur 4 caractères (non affichée au niveau du Finder). Par exemple, le type de fichier d'un document texte est "TEXT". De plus, sous MacOS, un document est identifié par un créateur, désignant l'application qui a créé le document. Cette notion n'existe pas sous Windows. Le créateur d'un document est déterminé par la propriété de type de fichier du document. Par exemple, le créateur de ce document créé avec 4D version 6 est "4D06".

DocRef : numéro de référence de document

Un document est **ouvert** ou **fermé**. Avec les commandes 4D, un document ne peut être ouvert que par un process à la fois. Un process peut ouvrir plusieurs documents, plusieurs process peuvent ouvrir de multiples documents, mais vous ne pouvez pas ouvrir le même document deux fois en même temps.

Vous ouvrez un document à l'aide des fonctions Ouvrir document, Creer document et Ajouter a document.
Une fois que le document est ouvert, vous pouvez lire et écrire des caractères dans ce document (cf. les commandes RECEVOIR PAQUET et ENVOYER PAQUET). Lorsque vous en avez terminé avec un document, il est préférable de le fermer — avec la commande FERMER DOCUMENT.

Tous les documents ouverts sont désignés au moyen de l'expression **DocRef**, retournée par les commandes Ouvrir document, Creer document et Ajouter a document. Une DocRef identifie de façon unique un document ouvert. C'est une expression de type Heure. Toutes les commandes fonctionnant avec des documents ouverts attendent une DocRef comme paramètre. Si vous passez une DocRef incorrecte à l'une de ces commandes, une erreur du gestionnaire de fichiers est générée.

Gestion des erreurs E/S

Quand vous accédez à des documents (ouverture, fermeture, suppression, changement de nom, copie), quand vous modifiez les propriétés d'un document ou quand vous lisez et écrivez des caractères dans un document, des erreurs d'entrée/sortie (E/S) peuvent se produire. Un document peut ne pas avoir été trouvé ; il peut être verrouillé ; il peut être déjà ouvert. Vous pouvez repérer ces erreurs grâce à une méthode de gestion des erreurs installée par la commande APPELER SUR ERREUR. La plupart des erreurs qui peuvent se produire lors de l'utilisation des commandes du thème documents système est décrite dans la section Erreurs du gestionnaire de fichiers du système.

La variable système Document

Les trois commandes Ouvrir document, Creer document et Ajouter a document vous permettent d'accéder à un document par les boîtes de dialogue standard d'ouverture ou d'enregistrement de fichiers. Quand vous accédez à un document par ces boîtes de dialogue standard, 4D retourne le chemin d'accès complet du document dans la variable système Document. Ne confondez pas cette variable système avec le paramètre document qui apparaît dans la liste des paramètres des commandes.

Spécification des noms et chemins d'accès des documents

La plupart des routines de cette section attendant un nom de document acceptent à la fois un nom et un chemin d'accès au document (*). Si vous passez un nom, la commande cherche le document dans le dossier de la base. Si vous passez un chemin d'accès, il doit être valide.

Si vous passez un nom ou un chemin d'accès incorrect, la commande génère une erreur du gestionnaire de fichiers que vous pouvez intercepter avec une méthode d'APPELER SUR ERREUR.

(*) sauf cas contraire spécifié explicitement.

Attention : La longueur maximale du paramètre document est de 255 caractères. Si vous passez un nom plus long, il sera tronqué et une erreur du gestionnaire de fichiers sera générée.

Méthodes projet utiles pour la gestion des documents sur disque

• Détecter sur quelle plate-forme vous opérez

Bien que 4e Dimension fournisse des commandes telles que ASSOCIER TYPES FICHIER destinées à éliminer les modifications de code liées aux particularités des plates-formes, lorsque vous commencez à travailler à un plus bas niveau en manipulant des documents sur disque, par exemple lorsque vous obtenez les chemins d'accès par programmation, vous avez besoin de savoir si vous fonctionnez sous Windows ou MacOS.

La méthode projet Sous Windows listée ci-dessous vous permet de savoir si votre base tourne sous Windows :

```
` Méthode projet Sous Windows
` Sous Windows -> Booléen
` Sous Windows -> Vrai si la base est sous Windows

C_BOOLEEN($0)
C_ENTIER LONG($vlPlatform;$vlSystem;$vlMachine)

PROPRIETES PLATE FORME($vlPlatform;$vlSystem;$vlMachine)
$0:=($vlPlatform=Windows)
```

• Utiliser le bon symbole séparateur de dossiers

Sous Windows, un niveau de dossier est symbolisé par une barre oblique inversée \. Sous MacOS, un niveau de dossier est symbolisé par deux-points :

En fonction de la plate-forme sur laquelle tourne la base, la méthode projet Symbole séparateur suivante vous retourne le code ASCII du caractère séparateur de dossiers.

```
` Méthode projet Symbole séparateur
` Symbole séparateur -> Entier
` Symbole séparateur -> code ASCII de "\" (Windows) ou ":" (MacOS)

C_ENTIER($0)
Si (Sous Windows )
   $0:=Code ascii("\")
Sinon
   $0:=Code ascii(":")
Fin de si
```

• Extraire le nom de fichier d'un chemin d'accès complet (ou "nom long")

Une fois que vous avez récupéré le "nom long" d'un fichier (c'est-à-dire le chemin d'accès+le nom du fichier), vous pouvez avoir besoin d'en extraire le nom du fichier seul, par exemple pour l'afficher comme titre d'une fenêtre. La méthode projet Nom long vers nom de fichier vous le permet, sous Windows et MacOS.

```
` Méthode projet Nom long vers nom de fichier
` Nom long vers nom de fichier ( Chaîne ) -> Chaîne
` Nom long vers nom de fichier ( nom long ) -> nom de fichier

C_ALPHA(255;$1;$0)
C_ENTIER($viLen;$viPos;$viChar;$viDirSymbol)
$viDirSymbol:=Symbole séparateur
$viLen:=Longueur($1)
$viPos:=0
Boucle ($viChar;$viLen;1;-1)
   Si (Code ascii($1≤$viChar≥)=$viDirSymbol)
      $viPos:=$viChar
      $viChar:=0
   Fin de si
Fin de boucle
Si ($viPos>0)
   $0:=Sous chaine($1;$viPos+1)
Sinon
   $0:=$1
Fin de si
Si (◊vbDebugOn)` Mettre la variable à Vrai ou Faux dans la méthode base Sur ouverture
   Si ($0="")
      TRACE
   Fin de si
Fin de si
```

• Extraire le chemin d'accès seul du chemin d'accès complet (ou "nom long")

Lorsque vous avez récupéré le "nom long" d'un fichier (c'est-à-dire le chemin d'accès+le nom du fichier), vous pouvez avoir besoin d'en extraire uniquement le chemin d'accès au fichier, par exemple pour sauvegarder d'autres documents au même endroit. La méthode Nom long vers chemin accès vous le permet, sous Windows et MacOS.

```
` Méthode projet Nom long vers chemin accès
` Nom long vers chemin accès ( Chaîne ) -> Chaîne
` Nom long vers chemin accès ( nom long ) -> chemin d'accès

C_ALPHA(255;$1;$0)
C_ALPHA(1;$vsDirSymbol)
C_ENTIER($viLen;$viPos;$viChar;$viDirSymbol)

$viDirSymbol:=Symbole séparateur
$viLen:=Longueur($1)
$viPos:=0
Boucle ($viChar;$viLen;1;-1)
   Si (Code ascii($1≤$viChar≥)=$viDirSymbol)
      $viPos:=$viChar
      $viChar:=0
   Fin de si
Fin de boucle
Si ($viPos>0)
   $0:=Sous chaine($1;1;$viPos)
Sinon
   $0:=$1
Fin de si
Si (◊vbDebugOn)` Mettre la variable à Vrai ou Faux dans la méthode base Sur ouverture
   Si ($0="")
      TRACE
   Fin de si
Fin de si
```

Référence

Ajouter a document, ASSOCIER TYPES FICHIER, CHANGER CREATEUR DOCUMENT, CHANGER POSITION DANS DOCUMENT, CHANGER PROPRIETES DOCUMENT, CHANGER TAILLE DOCUMENT, CHANGER TYPE DOCUMENT, COPIER DOCUMENT, Createur document, Creer document, CREER DOSSIER, DEPLACER DOCUMENT, FERMER DOCUMENT, LISTE DES DOCUMENTS, LISTE DES DOSSIERS, LISTE DES VOLUMES, Ouvrir document, Position dans document, PROPRIETES DOCUMENT, PROPRIETES DU VOLUME, SUPPRIMER DOCUMENT, Taille document, Tester chemin acces, Type document.

Type document (document) → Alpha

Paramètre	Type		Description
document	Alpha	→	Nom de document ou Chemin d'accès complet à un document
Résultat	Alpha	←	Extension de fichier Windows (chaîne de 1 à 3 caractères) ou type de fichier MacOS (chaîne de 4 caractères)

Description

La commande Type document retourne le type du document dont vous avez passé le nom ou le chemin d'accès dans document.

Sous Windows, Type document retourne l'extension de fichier du document (par exemple 'DOC' pour un document Microsoft Word, 'EXE' pour un fichier exécutable, etc.) ou le type de document MacOS (4 caractères) correspondant si ce dernier a été associé à une extension Windows équivalente par 4e Dimension (par exemple 'TEXT' pour l'extension 'TXT') ou par un appel antérieur à la commande ASSOCIER TYPES FICHIER.

Sous MacOS, Type document retourne le type de fichier MacOS (4 caractères) du document (par exemple 'TEXT' pour un document de type Texte, 'APPL' pour une application double-cliquable, etc.).

Référence

ASSOCIER TYPES FICHIER, CHANGER TYPE DOCUMENT, Createur document, PROPRIETES DOCUMENT.

CHANGER TYPE DOCUMENT

Documents système

version 6.0

CHANGER TYPE DOCUMENT (document; type)

Paramètre	Type		Description
document	Alpha	→	Nom de document ou Chemin d'accès complet à un document
type	Alpha	→	Extension de fichier Windows (chaîne de 1 à 3 caractères) ou Type de fichier MacOS (chaîne de 4 caractères)

Description

La commande CHANGER TYPE DOCUMENT définit le type du document dont vous avez passé le nom ou le chemin d'accès complet dans document.

Vous passez le nouveau type du document dans type. Reportez-vous à la description des types de documents dans les sections Présentation des documents système et Type document.

Sous Windows, la commande modifie l'extension et donc le nom du fichier décrit par document.
Par exemple, l'instruction CHANGER TYPE DOCUMENT("C:\Docs\Facture.asc";"TEXT") renomme le fichier "Facture.asc" en "Facture.txt" (dans 4D, le type Mac "TEXT" est associé au type Windows "txt").
Si le type n'est pas associé dans 4D, il suffit de passer directement les trois lettres de l'extension. Par exemple, l'instruction CHANGER TYPE DOCUMENT("C:\Docs\Facture.asc"; "zip") renomme le fichier "Facture.asc" en "Facture.zip".

Référence

ASSOCIER TYPES FICHIER, CHANGER CREATEUR DOCUMENT, CHANGER PROPRIETES DOCUMENT, Type document.

Createur document (document) → Alpha

Paramètre	Type		Description
document	Alpha	→	Nom de document ou Chemin d'accès complet à un document
Résultat	Alpha	←	Chaîne vide (Windows) ou Créateur de fichier (MacOS)

Description

La commande Createur document retourne le "créateur" du document dont vous avez passé le nom ou le chemin d'accès complet dans document.

Sous Windows, Createur document retourne une chaîne vide.

Référence

CHANGER CREATEUR DOCUMENT, Type document.

CHANGER CREATEUR DOCUMENT (document; créateur)

Paramètre	Type		Description
document	Alpha	→	Nom de document ou Chemin d'accès complet à un document
créateur	Alpha	→	Créateur de fichier (MacOS) ou Chaîne vide (Windows)

Description

La commande CHANGER CREATEUR DOCUMENT définit le créateur du document dont vous avez passé le nom ou le chemin d'accès complet dans document.

Vous passez le nouveau créateur du document dans créateur.

Sous Windows, cette commande ne fait rien.

Reportez-vous à la description du créateur de document dans la section Présentation des documents système.

Référence

CHANGER PROPRIETES DOCUMENT, CHANGER TYPE DOCUMENT, Createur document.

Ouvrir document

Documents système

version 6.5 (Modifiée)

Ouvrir document (document{; type{; mode}}) → DocRef

Paramètre	Type		Description
document	Alpha	→	Nom du document ou Chemin d'accès complet au document ou Chaîne vide pour afficher la boîte de dialogue
type	Alpha	→	Type de fichier MacOS (chaîne de 4 caractères) ou Extension de fichier Windows (chaîne de 1 à 3 caractères) ou fichier texte (.TXT) si omis
mode	Entier	→	Mode d'ouverture du document
Résultat	DocRef	←	Numéro de référence du document

Description

La commande Ouvrir document ouvre le document dont vous avez passé le nom dans document.

Si vous passez une chaîne vide ("") dans document, une boîte de dialogue standard d'ouverture de fichiers apparaît et l'utilisateur peut désigner le document. Si dans ce cas l'utilisateur clique sur le bouton **Annuler**, aucun document n'est ouvert, Ouvrir document retourne une référence de document nulle, et la variable OK prend la valeur 0.

Si le document est correctement ouvert, Ouvrir document retourne sa référence de document et la variable OK prend la valeur 1. Si le document n'existe pas ou est déjà ouvert, une erreur est générée.

Sous MacOS, si vous appelez la boîte de dialogue standard d'ouverture de fichiers, tous les types de documents sont affichés par défaut. Pour afficher un type spécifique de document, passez un type dans le paramètre optionnel type.

Sous Windows, si vous appelez la boîte de dialogue standard d'ouverture de fichiers, tous les documents *.* sont affichés par défaut. Pour afficher d'autres types de documents, passez une extension de fichier Windows (de 1 à 3 caractères) ou un type de fichier MacOS associé à l'aide de la commande ASSOCIER TYPES FICHIER dans le paramètre optionnel type.

Sous Windows, même si vous n'appelez pas la boîte de dialogue standard d'ouverture de fichiers, vous pouvez passer une valeur dans type pour spécifier le type de document à ouvrir. Par défaut, Ouvrir document tente d'ouvrir un fichier .TXT. Si vous passez le paramètre type, Ouvrir document tentera d'ouvrir le document dont le nom est "document.type". Exemple :

⇒ vhRefDoc:=**Ouvrir document**("C:\Lettre";"WRI")

Dans ce cas, 4e Dimension recherche le document "C:\Lettre.WRI" sur le disque dur. Si vous passez plus de trois caractères dans type, Ouvrir document prend en compte seulement les trois premiers caractères. Si aucun type de document n'est spécifié, 4e Dimension essaie d'ouvrir le document sans extension de fichier. S'il ne le trouve pas, il essaie d'ouvrir le document avec l'extension .TXT. S'il ne le trouve pas, l'erreur "Fichier non trouvé" est affichée.

Lorsqu'un document est ouvert, Ouvrir document se place initialement au début du document, alors que Ajouter a document se place à la fin.

Une fois que vous avez ouvert un document, vous pouvez écrire ou lire des valeurs dans ce document à l'aide des commandes RECEVOIR PAQUET et ENVOYER PAQUET, que vous pouvez combiner avec les commandes Position dans document et CHANGER POSITION DANS DOCUMENT pour accéder directement à certains endroits du document.

Le paramètre optionnel mode permet de définir le mode d'ouverture du fichier document. Quatre modes d'ouverture sont disponibles. 4e Dimension vous propose les constantes prédéfinies suivantes, placées dans le thème Documents systèmes :

Constante	**Type**	**Valeur**
Lecture et écriture (valeur par défaut)	Entier	0
Mode écriture	Entier	1
Mode lecture	Entier	2
Lire chemin accès	Entier	3

N'oubliez pas d'appeler finalement FERMER DOCUMENT pour le document.

Exemples

(1) L'exemple suivant ouvre un document existant qui s'appelle “Note”, écrit la chaîne “Au revoir” dans le document et le referme. Si le document contient déjà la chaîne “Bonjour”, elle est remplacée :

```
      C_HEURE(vDoc)
⇒     vDoc := Ouvrir document ("Note")  ` Ouvrir le document Note
      Si (OK=1)
         ENVOYER PAQUET (vDoc; "Au revoir")  ` Ecrire un mot dans le document
         FERMER DOCUMENT (vDoc)  ` Fermer le document
      Fin de si
```

(2) Vous pouvez lire un document déjà ouvert en écriture :

```
⇒     vDoc:=Ouvrir document ("PassFile";"TEXT")     ` Le fichier est ouvert
⇒     vRef:=Ouvrir document ("PassFile";"TEXT";Mode lecture)  ` Le fichier est lu
```

Variables et ensembles système

Si le document est correctement ouvert, la variable système *OK* prend la valeur 1, sinon elle prend la valeur 0. Après l'appel, la variable système *Document* contient le nom complet du document.
Si vous passez la valeur 3 dans mode, la fonction retourne ?00:00:00? (pas de référence de document). Le document n’est pas ouvert mais les variables système *Document* et *OK* sont mises à jour :
• *OK* prend la valeur 1,
• *Document* contient soit le nom, soit le chemin d’accès et le nom du fichier document, suivant la valeur passée dans document (si vous avez passé un nom de fichier, *Document* contiendra ce nom, si vous avez passé un chemin d’accès complet, *Document* contiendra ce chemin d’accès complet).

Note : Si le fichier désigné par document n’est pas trouvé ou si vous passez une chaîne vide dans document, une boîte de dialogue d’ouverture de fichiers apparaît. Si elle est validée, *Document* et *OK* sont mises à jour comme décrit ci-dessus. Si elle est annulée, *OK* prend la valeur 0.

Référence

Ajouter a document, Creer document.

Creer document (document{; type}) → DocRef

Paramètre	Type		Description
document	Alpha	→	Nom de document ou Chemin d'accès complet de document ou Chaîne vide pour afficher la boîte de dialogue standard d'ouverture de fichiers
type	Alpha	→	Type de fichier MacOS (chaîne de 4 caractères) ou Extension de fichier Windows (chaîne de 1 à 3 caractères) ou Document texte (.TXT) si omis
Résultat	docRéf	←	Numéro de référence du document

Description

La commande Creer document crée un document et retourne son numéro de référence de document.

Vous passez le nom ou le chemin d'accès complet du nouveau document dans document. Si document existe déjà, il est remplacé. Cependant, si le document est verrouillé ou est déjà ouvert, une erreur est générée.

Si vous passez une chaîne vide dans document, une boîte de dialogue standard d'enregistrement de fichiers apparaît et l'utilisateur peut spécifier le nom du document. Si dans ce cas l'utilisateur clique sur le bouton **Annuler**, Creer document retourne une référence de document nulle, et la variable OK prend la valeur 0.

Si le document est correctement créé et ouvert, Creer document retourne sa référence de document et la variable OK prend la valeur 1. Pour créer un autre type de document, passez un type dans le paramètre optionnel type.

Que vous utilisiez ou non la boîte de dialogue standard d'enregistrement de fichiers, Creer document crée par défaut un document de type TEXT (MacOS) ou .TXT (Windows).

Sous MacOS, vous passez un type de fichier. Sous Windows, vous passez une extension de fichier Windows (de 1 à 3 caractères) ou un type de fichier MacOS associé à l'aide de la commande ASSOCIER TYPES FICHIER.

Une fois que vous avez créé et ouvert un document, vous pouvez écrire ou lire des valeurs dans ce document à l'aide des commandes RECEVOIR PAQUET et ENVOYER PAQUET, que vous pouvez combiner avec les commandes Position dans document et CHANGER POSITION DANS DOCUMENT pour accéder directement à certains endroits du document.

N'oubliez pas d'appeler finalement FERMER DOCUMENT pour le document.

Exemple

L'exemple suivant crée et ouvre un nouveau document qui s'appelle “Note”, écrit la chaîne “Bonjour” et le referme :

```
      C_HEURE(vDoc)
⇒     vDoc := Creer document ("Note")  ` Créer un nouveau document qui s'appelle Note
      Si (OK=1)
         ENVOYER PAQUET (vDoc; "Bonjour")  ` Ecrire un mot dans le document
         FERMER DOCUMENT (vDoc)  ` Fermer le document
      Fin de si
```

Référence

Ajouter a document, Ouvrir document.

Ajouter a document (document{; type}) → DocRef

Paramètre	Type		Description
document	Alpha	→	Nom du document ou Chemin d'accès complet au document ou Chaîne vide pour afficher le dialogue standard d'ouverture de fichiers
type	Alpha	→	Type de fichier MacOS (chaîne de 4 caractères) ou Extension de fichier Windows (chaîne de 1 à 3 caractères) ou Fichier texte (.TXT) si omis
Résultat	DocRef	←	Numéro de référence du document

Description

La commande Ajouter a document "fait la même chose" que la commande Ouvrir document : elle vous permet d'ouvrir un document sur disque.
La seule différence est que Ajouter a document se place initialement à la fin du document, alors que Ouvrir document se place au début.

Pour plus d'informations, reportez-vous à la description de la commande Ouvrir document.

Exemple

L'exemple suivant ouvre un document existant qui s'appelle “Note”, ajoute à la fin du document la chaîne “ et à bientôt” suivie d'un retour chariot puis le referme. Si le document contenait déjà la chaîne “Au revoir”, il contiendra la chaîne “Au revoir et à bientôt” suivie d'un retour chariot :

```
    C_HEURE(vDoc)
⇒   vDoc := Ajouter a document ("Note")  ` Ouvrir le document Note
    ENVOYER PAQUET (vDoc; " et à bientôt" + Caractere (13))  ` Ajouter la chaîne
    FERMER DOCUMENT (vDoc)  ` Fermer le document
```

Variables et ensembles système

Si le document est correctement ouvert, la variable système OK prend la valeur 1, sinon elle prend la valeur 0. La variable système Document contient le nom du document.

Référence

Creer document, Ouvrir document.

FERMER DOCUMENT (réfDocument)

Paramètre	Type		Description
réfDocument	DocRef	→	Numéro de référence du document

Description

FERMER DOCUMENT ferme le document spécifié par réfDocument.

Fermer un document est le seul moyen de s'assurer que les données écrites dans le fichier sont sauvegardées. Vous devez fermer tous les documents ouverts par les commandes Ouvrir document, Creer document et Ajouter a document.

Exemple

L'exemple suivant permet à l'utilisateur de créer un nouveau document, écrit la chaîne “Bonjour”, puis le referme :

```
      C_HEURE(vDoc)
      vDoc := Creer document ("")
      Si (OK=1)
         ENVOYER PAQUET (vDoc; "Bonjour")  ` Ecrire un mot dans le document
⇒        FERMER DOCUMENT (vDoc)  ` Fermer le document
      Fin de si
```

Référence

Ajouter a document, Creer document, Ouvrir document.

COPIER DOCUMENT (nomSource; nomDest{; *})

Paramètre	Type		Description
nomSource	Alpha	→	Nom du document à copier
nomDest	Alpha	→	Nom du document copié
*		→	Remplacer document existant le cas échéant

Description

La commande COPIER DOCUMENT copie le document désigné par nomSource à l'emplacement désigné par nomDest.

Les deux paramètres nomSource et nomDest peuvent désigner un document situé dans le dossier de la base, ou un chemin d'accès complet exprimé par rapport à la racine du volume.

Une erreur est générée si un document nommé nomDest existe déjà, à moins que vous n'ayez spécifié le paramètre optionnel *, qui indique à COPIER DOCUMENT de supprimer et de remplacer le document existant par le document de destination dans ce cas.

Exemples

(1) L'exemple suivant duplique un document dans son propre dossier :

⇒ **COPIER DOCUMENT**("C:\DOSSIER\LeDoc";"C:\DOSSIER\LeDoc2")

(2) L'exemple suivant copie un document dans le dossier de la base (dans la mesure où C:\DOSSIER n'est pas le dossier de la base) :

⇒ **COPIER DOCUMENT**("C:\DOSSIER\LeDoc";"LeDoc")

(3) L'exemple suivant copie un document d'un volume vers un autre :

⇒ **COPIER DOCUMENT**("C:\DOSSIER\LeDoc";"F:\Archives\LeDoc.OLD")

(4) L'exemple suivant duplique un document dans son propre dossier, écrasant la précédente copie si elle existe :

⇒ **COPIER DOCUMENT**("C:\DOSSIER\LeDoc";"C:\DOSSIER\LeDoc2";*)

Référence

DEPLACER DOCUMENT.

DEPLACER DOCUMENT (cheminSource; cheminDest)

Paramètre	Type		Description
cheminSource	Alpha	→	Chemin d'accès complet au document existant
cheminDest	Alpha	→	Chemin d'accès de destination

Description

La commande DEPLACER DOCUMENT déplace ou renomme un document.

Vous passez le chemin d'accès complet au document existant dans le paramètre cheminSource et le nouveau nom et/ou emplacement du document dans cheminDest.

Attention : Avec DEPLACER DOCUMENT, vous pouvez déplacer un document depuis et vers tous les dossiers du même volume. Si vous souhaitez déplacer un document entre deux volumes différents, utilisez la commande COPIER DOCUMENT pour “déplacer” le document puis effacez le document original avec la commande SUPPRIMER DOCUMENT.

Exemples

(1) L'exemple suivant renomme le document DocNom :

```
⇒ DEPLACER DOCUMENT("C:\DOSSIER\DocNom";"C:\DOSSIER\NouveauDocNom")
```

(2) L'exemple suivant déplace et renomme le document DocNom :

```
⇒ DEPLACER DOCUMENT("C:\DOSSIER1\DocNom";"C:\DOSSIER2\NouveauDocNom")
```

(3) L'exemple suivant déplace le document DocNom :

```
⇒ DEPLACER DOCUMENT("C:\DOSSIER1\DocNom";"C:\DOSSIER2\DocNom")
```

Note : Dans les deux derniers exemples, le dossier de destination "C:\DOSSIER2" doit déjà exister. En effet, la commande DEPLACER DOCUMENT déplace uniquement un document, elle ne peut créer de dossiers.

Référence

COPIER DOCUMENT.

SUPPRIMER DOCUMENT (document)

Paramètre	Type		Description
document	Alpha	→	Nom de document ou Chemin d'accès complet au document

Description

SUPPRIMER DOCUMENT supprime le document dont vous avez passé le nom dans document.

Si le document n'existe pas, aucune erreur n'est générée. En revanche, un erreur est retournée si vous tentez de supprimer un document ouvert.

SUPPRIMER DOCUMENT n'accepte pas de chaîne vide dans le paramètre document. Si vous passez une chaîne vide, la boîte de dialogue standard d'ouverture de fichiers ne s'affiche pas et une erreur est générée.

ATTENTION : SUPPRIMER DOCUMENT peut supprimer tout fichier disque, y compris des fichiers créés par d'autres applications ou les applications elles-mêmes. La commande SUPPRIMER DOCUMENT doit donc être utilisée avec précaution. La suppression d'un document est une opération définitive et irréversible.

Exemples

(1) L'exemple suivant supprime le document appelé Note :

```
⇒ SUPPRIMER DOCUMENT ("Note") ` Suppression du document
```

(2) Reportez-vous à l'exemple de la commande AJOUTER A PRESSE PAPIERS.

Variables et ensembles système

La suppression d'un document met la variable système OK à 1. Si SUPPRIMER DOCUMENT ne peut pas supprimer le document, la variable système OK prend la valeur 0.

Tester chemin acces (cheminAccès) → Numérique

Paramètre	Type		Description
cheminAccès	Alpha	→	Chemin d'accès à un dossier ou un document
Résultat	Numérique	←	1= cheminAccès est un document existant 0 = cheminAccès est un dossier existant <0 = chemin d'accès invalide, code d'erreur du gestionnaire de fichiers du système

Description

La fonction Tester chemin acces vérifie si le document ou le dossier dont vous avez passé le chemin d'accès et le nom dans cheminAccès est présent sur le disque.
Si un document est trouvé, Tester chemin acces retourne 1. Si un dossier est trouvé, Tester chemin acces retourne 0.

4D propose les constantes prédéfinies suivantes :

Constante	Type	Valeur
Est un document	Entier long	1
Est un dossier	Entier long	0

Si aucun document ou dossier n'est trouvé, Tester chemin acces retourne une valeur négative (par exemple -43 pour "Fichier non trouvé").

Exemple

L'exemple suivant teste la présence du document “Journal” dans le dossier de la base et le crée s'il n'existe pas:

```
⇒   Si (Tester chemin acces("Journal") # Est un document)
       $vhDocRef:=Creer document("Journal")
       Si (OK=1)
          FERMER DOCUMENT($vhDocRef)
       Fin de si
    Fin de si
```

Référence

Creer document, CREER DOSSIER.

CREER DOSSIER (cheminAccès)

Paramètre	Type		Description
cheminAccès	Alpha	→	Chemin d'accès au nouveau dossier à créer

Description

La commande CREER DOSSIER crée un dossier en fonction du chemin d'accès que vous passez dans le paramètre cheminAccès.

Si vous passez un nom, le dossier est créé dans le dossier de la base. Si vous passez un chemin d'accès complet, cela doit être un chemin d'accès valide au nom du dossier à créer, à partir de la racine du volume ou du dossier de la base.

Exemples

(1) L'exemple suivant crée le dossier “Archives” dans le dossier de la base :

```
⇒ CREER DOSSIER("Archives")
```

(2) L'exemple suivant crée le dossier “Archives” dans le dossier de la base, puis crée les sous-dossiers “Janvier” et “Février”:

```
⇒ CREER DOSSIER("Archives")
⇒ CREER DOSSIER("Archives\Janvier")
⇒ CREER DOSSIER("Archives\Février")
```

(3) L'exemple suivant crée le dossier “Archives” à la racine du volume C :

```
⇒ CREER DOSSIER("C:\Archives")
```

(4) L'exemple suivant échouera s'il n'existe pas déjà de dossier “Nouveautés” à la racine du volume C :

```
⇒ CREER DOSSIER("C:\Nouveautés\Images")
    ` INCORRECT, on ne peut pas créer deux niveaux de dossier en un seul appel
```

Référence

LISTE DES DOSSIERS, Selectionner dossier, Tester chemin acces.

Selectionner dossier {(message)} → Alpha

Paramètre	Type		Description
message	Alpha	→	Titre de la fenêtre de sélection
Résultat	Alpha	←	Chemin d'accès au dossier sélectionné

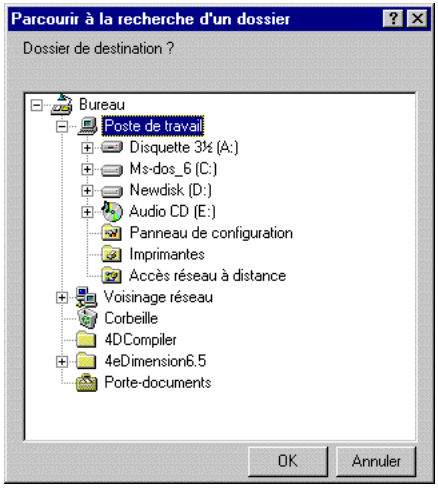
Description

La commande Selectionner dossier affiche une boîte de dialogue permettant de désigner manuellement un dossier, et de récupérer en retour de fonction le chemin d'accès complet au dossier sélectionné.

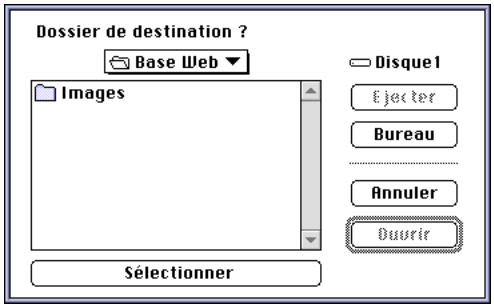
Note : Cette commande ne modifie pas le dossier courant de l'application 4D.

La commande Selectionner dossier affiche une boîte de dialogue standard de navigation à travers les volumes et les dossiers du poste.
Le paramètre optionnel message permet d'afficher une ligne d'information dans la boîte de dialogue (dans notre exemple, message a pour valeur "Dossier de destination ?").

• *Windows* :



• *MacOS* :

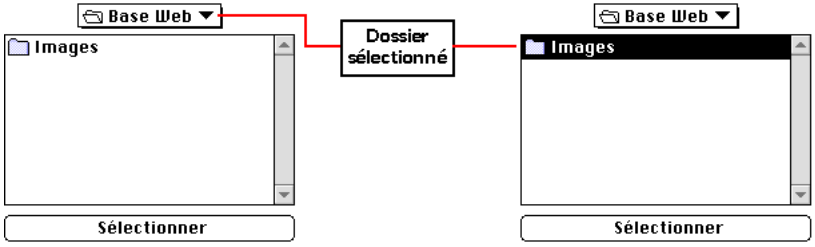


L'utilisateur sélectionne un dossier en cliquant sur le bouton **OK** (Windows) ou **Sélectionner** (MacOS). Le chemin d'accès au dossier choisi est alors retourné par la fonction.

• Sous Windows, la chaîne retournée est du type :
"C:\Dossier1\Dossier2\DossierSélectionné\"

• Sous MacOS, la chaîne retournée est du type :
"Disque:Dossier1:Dossier2:DossierSélectionné:"

Note MacOS : Sous MacOS, selon que le nom du dossier est sélectionné ou non dans la boîte de dialogue, le chemin retourné est différent.



4D Server : Cette fonction permet de visualiser les volumes connectés aux postes clients. Il n'est pas possible de l'appeler depuis une procédure stockée.

Si l'utilisateur clique sur le bouton de sélection, la variable système *OK* prend la valeur 1. Si l'utilisateur clique sur le bouton d'annulation, *OK* prend la valeur 0 et la fonction retourne une chaîne vide.

Note : Sous Windows, si l'utilisateur a sélectionné certains éléments incorrects tels que "Poste de travail", "Corbeille", etc., la variable système *OK* prend la valeur 0, même si la boîte de dialogue est validée.

Exemple

L'exemple suivant permet de sélectionner le dossier dans lequel toutes les images de la bibliothèque d'images seront enregistrées :

```
⇒   $DossierImages:=Selectionner dossier("Sélectionnez un dossier pour vos images.")
    LISTE IMAGES DANS BIBLIOTHEQUE (pictRefs;pictNoms)
    Boucle ($n;1;Taille tableau(pictNames))
       $vRef:=Creer document($DossierImages+pictNoms{$n};"PICT")
       Si (OK=1)
          LIRE IMAGE DANS BIBLIOTHEQUE(pictRefs{$n};$vPictSauvegarde)
          ENREGISTRER IMAGE($vRef;$vPictSauvegarde)
          FERMER DOCUMENT($vRef)
       Fin de si
    Fin de boucle
```

Référence

CREER DOSSIER, LISTE DES DOSSIERS.

Variables et ensembles système

Si l'utilisateur clique sur le bouton de sélection, la variable système *OK* prend la valeur 1. Si l'utilisateur clique sur le bouton d'annulation, *OK* prend la valeur 0.

SUPPRIMER DOSSIER (dossier)

Paramètre	Type		Description
dossier	Alpha	→	Nom ou chemin d'accès complet du dossier à supprimer

Description

La commande SUPPRIMER DOSSIER supprime le dossier dont vous avez passé le nom ou le chemin d'accès complet dans dossier.

Seuls les dossiers vides peuvent être supprimés par cette commande.
• Si vous tentez de supprimer un dossier contenant des éléments, l'erreur -47 (Tentative de suppression d'un dossier non vide) est générée.
• Si vous passez dans dossier le chemin d'accès d'un fichier, ou une chaîne vide, ou encore le chemin d'accès d'un dossier inexistant, la commande ne fait rien et génère l'erreur -43 (Fichier non trouvé).

Vous pouvez intercepter ces erreurs à l'aide d'une méthode installée par la commande APPELER SUR ERREUR.

Référence

SUPPRIMER DOCUMENT.

CREER ALIAS (cheminCible; cheminAlias)

Paramètre	Type		Description
cheminCible	Alpha	→	Nom ou chemin d'accès de la cible de l'alias/du raccourci
cheminAlias	Alpha	→	Nom ou chemin d'accès complet de l'alias/du raccourci à créer

Description

La commande CREER ALIAS crée un alias (appelé "raccourci" sous Windows) du fichier ou dossier cible désigné par le paramètre cheminCible, avec le nom et l'emplacement définis dans le paramètre cheminAlias.

Vous pouvez créer un alias de tout type de document ou de dossier. L'icône de l'alias sera identique à celle de l'élément cible. Elle comportera en outre une petite flèche et, sous MacOS, le libellé de l'alias apparaîtra en caractères italiques.

La commande n'affecte pas de libellé par défaut à l'alias, vous devez passer un nom dans le paramètre cheminAlias. Si vous passez uniquement un nom dans ce paramètre, l'alias est créé dans le dossier actif courant (généralement, le dossier contenant le fichier de structure de la base).

Note : Sous Windows, les raccourcis sont des fichiers dont l'extension est ".LNK". Si vous ne passez pas cette extension, la commande l'ajoute automatiquement.

Si vous passez une chaîne vide dans cheminCible, la commande ne fait rien.

Exemple

Votre base génère des fichiers texte intitulés "Rapport*Numéro*", stockés dans le dossier de la base. Vous souhaitez permettre à l'utilisateur de créer des raccourcis vers ces rapports et de les stocker où il le souhaite :

```
   `Méthode CREER_RAPPORT
C_TEXTE($vtRapport)
C_ALPHA(250;$vtChemin)
C_ALPHA(80;$vaNom)
C_HEURE(vDoc)
C_ENTIER($NumRapport)

$vTRapport:=...   `Edition du rapport
$NumRapport:=...   `Calcul du numéro du rapport
$vaNom:="Rapport"+Chaine($NumRapport)+".txt"   `Nom du fichier
vDoc:=Creer document($vaNom)
```

```
Si (OK=1)
   ENVOYER PAQUET(vDoc;$vTRapport)
   FERMER DOCUMENT(vDoc)
      `Ajout de l'alias
   CONFIRMER("Créer un alias pour ce rapport ?)
   Si (OK=1)
      $vtChemin:=Selectionner dossier ("Où souhaitez-vous créer l'alias ?")
      Si (OK=1)
⇒        CREER ALIAS ($vaNom;$vtChemin+$vaNom)
      Fin de si
   Fin de si
Fin de si
```

Référence

RESOUDRE ALIAS.

RESOUDRE ALIAS (cheminAlias; cheminCible)

Paramètre	Type		Description
cheminAlias	Alpha	→	Nom ou chemin d'accès complet de l'alias/ du raccourci
cheminCible	Alpha	←	Nom ou chemin d'accès complet de la cible de l'alias/du raccourci

Description

La commande RESOUDRE ALIAS retourne le chemin d'accès complet du fichier ou dossier cible d'un alias (appelé "raccourci" sous Windows).

Vous passez dans cheminAlias le nom ou le chemin d'accès complet de l'alias.

Après l'exécution de la commande, la variable cheminCible contient le chemin d'accès complet du fichier ou dossier cible de l'alias.

Note : Sous Windows, les raccourcis sont des fichiers dont l'extension est ".LNK". Si vous ne passez pas cette extension, la commande l'ajoute automatiquement.

Référence

CREER ALIAS.

LISTE DES VOLUMES (volumes)

Paramètre	Type		Description
volumes	Tableau	←	Noms des volumes actuellement montés

Description

LISTE DES VOLUMES remplit le tableau volumes, de type Texte ou Alpha, avec les noms des volumes définis (Windows) ou montés (MacOS) sur votre machine.

Sous MacOS, elle retourne la liste des volumes visibles au niveau du Finder.

En revanche, sous Windows, elle retourne la liste des volumes couramment définis, même si le volume n'est pas physiquement présent (par exemple le volume "A:\" sera retourné même s'il n'y a pas de disquette dans le lecteur).

Exemple

A l'aide de la zone de défilement taVolumes, vous voulez afficher la liste des volumes définis ou montés sur votre machine :

```
   Au cas ou
      : (Evenement formulaire=Sur chargement)
         TABLEAU ALPHA(31;taVolumes;0)
⇒        LISTE DES VOLUMES(taVolumes)
         ` ...
   Fin de cas
```

Référence

LISTE DES DOCUMENTS, LISTE DES DOSSIERS, PROPRIETES DU VOLUME.

PROPRIETES DU VOLUME (volume; taille; utilisé; libre)

Paramètre	Type		Description
volume	Alpha	→	Nom du volume
taille	Numérique	←	Taille du volume exprimée en octets
utilisé	Numérique	←	Place utilisée sur le volume exprimée en octets
libre	Numérique	←	Place libre sur le volume exprimée en octets

Description

La commande PROPRIETES DU VOLUME retourne la taille, la place utilisée et la place libre sur le volume dont le nom est passé dans volume. Ces valeurs sont exprimées en octets.

Exemple

Votre application comprend des opérations par lots qui sont exécutées la nuit ou pendant le week-end. Ces opérations stockent des fichiers temporaires sur disque. Pour que cette méthode soit aussi autonome et souple que possible, vous écrivez une routine qui va automatiquement chercher et utiliser le premier volume ayant de la place disponible pour les fichiers temporaires. Voici la méthode :

```
   ` Méthode projet Chercher volume pour place
   ` Chercher volume pour place ( Reel ) -> Alpha
   ` Chercher volume pour place ( Place nécessaire en octets ) -> Nom du volume
                                                                  ou chaîne vide

C_ALPHA(31;$0)
C_ALPHA(255;$vaNomDoc)
C_ENTIER LONG($vlNbVolumes;$vlVolume)
C_REEL($1;$vlTaille;$vlUtilisé;$vlLibre)
C_HEURE($vhDocRef)

   ` Initialiser le résultat de la fonction
$0:=""
   ` Protéger les opérations d'entrée/sortie par une méthode d'interception d'erreur
APPELER SUR ERREUR("METHODE ERREUR")
   ` Obtenir la liste des volumes
TABLEAU ALPHA(31;$taVolumes;0)
gErreur:=0
```

```
LISTE DES VOLUMES($taVolumes)
Si (gErreur=0)
      ` Si nous sommes sous Windows, ignorer les deux lecteurs de disquettes
   Si (Sous Windows )
      $vlVolume:=Chercher dans tableau($taVolumes;"A:\")
      Si ($vlVolume>0)
         SUPPRIMER LIGNES($taVolumes;$vlVolume)
      Fin de si
      $vlVolume:=Chercher dans tableau($taVolumes;"B:\")
      Si ($vlVolume>0)
         SUPPRIMER LIGNES($taVolumes;$vlVolume)
      Fin de si
   Fin de si
   $vlNbVolumes:=Taille tableau($taVolumes)
      ` Pour chaque volume
   Boucle ($vlVolume;1;$vlNbVolumes)
         ` Obtenir la taille, la place utilisée et la place libre
      gErreur:=0
⇒     PROPRIETES DU VOLUME($taVolumes{$vlVolume};$vlTaille;$vlUtilisé;$vlLibre)
      Si (gErreur=0)
            ` Est-ce que la place libre est suffisante (plus 32K) ?
         Si ($vlLibre>=($1+32768))
               ` Si oui, vérifier que le volume n'est pas verrouillé...
            $vaNomDoc:=$taVolumes{$vlVolume}+Caractere(Symbole séparateur )
                                                +"XYZ"+Chaine(Hasard)+".TXT"
            $vhDocRef:=Creer document($vaNomDoc)
            Si (OK=1)
               FERMER DOCUMENT($vhDocRef)
               SUPPRIMER DOCUMENT($vaNomDoc)
                  ` Si tout est ok, retourner le nom du volume
               $0:=$taVolumes{$vlVolume}
               $vlVolume:=$vlNbVolumes+1
            Fin de si
         Fin de si
      Fin de si
   Fin de boucle
Fin de si
APPELER SUR ERREUR("")
```

Lorsque cette méthode projet est ajoutée à votre application, vous pouvez écrire :

```
$vaVolume:=Chercher volume pour place (100*1024*1024)
Si ($vaVolume#"")
   ` Continuer
Sinon
   ALERTE("Un volume avec au moins 100 Mo d'espace libre est nécessaire !")
Fin de si
```

Référence

LISTE DES VOLUMES.

LISTE DES DOSSIERS (cheminAccès; dossiers)

Paramètre	Type		Description
cheminAccès	Alpha	→	Chemin d'accès de volume, répertoire ou dossier
dossiers	Tableau	←	Noms des dossiers situés à cet endroit

Description

La commande LISTE DES DOSSIERS remplit le tableau de type Texte ou Alpha dossiers avec les noms des dossiers (répertoires sous Windows) situés à l'endroit que vous avez indiqué avec le paramètre cheminAccès.

S'il n'y pas de dossier à cet endroit, la commande retourne un tableau vide. Si le chemin d'accès que vous avez passé dans cheminAccès est invalide, LISTE DES DOSSIERS génère une erreur de gestionnaire de fichiers que vous pouvez intercepter à l'aide d'une méthode installée par APPELER SUR ERREUR.

ATTENTION : La longueur maximum du paramètre cheminAccès est de 255 caractères. Si vous passez un chemin d'accès avec un nombre de caractères supérieur à 255, il sera tronqué et une erreur de gestionnaire de fichier sera générée.

Référence

LISTE DES DOCUMENTS, LISTE DES VOLUMES, Selectionner dossier.

LISTE DES DOCUMENTS (cheminAccès; documents)

Paramètre	Type		Description
cheminAccès	Alpha	→	Chemin d'accès de volume ou de dossier
documents	Tableau	←	Nom des documents situés à cet endroit

Description

La commande LISTE DES DOCUMENTS remplit le tableau de type Texte ou Alpha documents avec les noms des documents situés à l'endroit que vous avez indiqué avec le paramètre cheminAccès.

S'il n'y pas de document à cet endroit, la commande retourne un tableau vide. Si le chemin d'accès que vous avez passé dans cheminAccès est invalide, LISTE DES DOCUMENTS génère une erreur de gestionnaire de fichier que vous pouvez intercepter à l'aide d'une méthode installée par APPELER SUR ERREUR.

Attention : La longueur maximum du paramètre cheminAccès est 255 caractères. Si vous passez un chemin d'accès avec un nombre de caractères supérieur à 255, il sera tronqué et une erreur de gestionnaire de fichiers du système est générée.

Référence

LISTE DES DOSSIERS, LISTE DES VOLUMES.

ASSOCIER TYPES FICHIER (macOS; windows; contexte)

Paramètre	Type		Description
macOS	Alpha	→	Type de fichier MacOS (4 caractères)
windows	Alpha	→	Extension de fichier Windows (1 à 3 caractères)
contexte	Alpha	→	Chaîne affichée dans la liste déroulante des types de fichiers dans la boîte de dialogue d'ouverture de fichiers sous Windows

Description

ASSOCIER TYPES FICHIER vous permet d'associer une extension de fichier Windows à un type de fichier MacOS.

Il suffit d'appeler cette routine une fois seulement pour associer des types de fichier dans une session de travail entière avec une base. Une fois que vous l'avez appelée, les commandes Ajouter a document, Creer document, Creer fichier ressources et Ouvrir fichier ressources, lorsqu'elle sont exécutées sous Windows, vont automatiquement substituer l'extension de fichier Windows au type de fichier MacOS que vous avez passé en paramètre à cette routine.

Dans le paramètre macOS, passez un type de fichier Macintosh de 4 caractères. Si vous ne passez pas une chaîne valide représentant un type de fichier, la commande ne fait rien et génère une erreur.

Dans le paramètre windows, passez une extension de fichier Windows de 1 à 3 caractères. Si vous ne passez pas une chaîne valide représentant une extension de fichier, la commande ne fait rien et génère une erreur.

Dans le paramètre contexte, passez la chaîne qui sera affichée dans le menu déroulant des types de fichiers dans la boîte de dialogue d'ouverture de fichiers sous Windows. La chaîne contexte est limitée à 32 caractères ; tout caractère supplémentaire est ignoré.

IMPORTANT: Une fois que vous avez associé une extension de fichier Windows à un type de fichier MacOS, vous ne pouvez pas modifier ou supprimer cette association dans la même session de travail. Si vous avez besoin de modifier l'association pendant le développement d'une application 4D, il faut réouvrir la base et exécuter de nouveau la commande.

Exemple

La ligne de code suivante (elle peut faire partie de la méthode base Sur ouverture) associe les fichiers de type MS-Word (sous MacOS “WDBN”) à l'extension de fichier Windows “.DOC” :

⇒ **ASSOCIER TYPES FICHIER** ("WDBN";"DOC";"Documents Word")

Une fois cette ligne exécutée, le code suivant n'affiche que des documents Word dans la boîte de dialogue d'ouverture de fichiers sous Windows et MacOS :

```
$DocRéf:=Ouvrir document("";"WDBN")
Si (OK=1)
   ...
Fin de si
```

Référence

Ajouter a document, Creer document, Creer fichier ressources, Ouvrir fichier ressources, Ouvrir fichier ressources.

PROPRIETES DOCUMENT (document; verrouillé; invisible; créé le; créé à ; modifié le; modifié à)

Paramètre	Type		Description
document	Alpha	→	Nom du document
verrouillé	Booléen	←	Verrouillé (Vrai) ou non verrouillé (Faux)
invisible	Booléen	←	Invisible (Vrai) ou visible (Faux)
créé le	Date	←	Date de création
créé à	Heure	←	Heure de création
modifié le	Date	←	Date de la dernière modification
modifié à	Heure	←	Heure de la dernière modification

Description

La commande PROPRIETES DOCUMENT retourne des informations sur le document dont le nom ou le chemin d'accès est passé dans le paramètre document.

Après l'appel :
• verrouillé retourne Vrai si le document est verrouillé. Un document verrouillé ne peut pas être ouvert ni supprimé.
• invisible retourne Vrai si le document est caché.
• créé le et créé à retournent la date et l'heure de création du document.
• modifié le et modifié à retournent la date et l'heure de la dernière modification du document.

Exemple

Vous avez créé une base de documentation et vous voulez exporter tous les enregistrements créés dans le base vers un document sur disque. Comme la base est régulièrement mise à jour, vous voulez écrire un algorithme d'export qui crée ou recrée chaque document sur disque si le document n'existe pas ou si l'enregistrement correspondant a été modifié depuis la dernière sauvegarde du document. Par conséquent, vous devez comparer la date et l'heure de modification du document (s'il existe) avec celles de l'enregistrement correspondant.

Pour illustrer cet exemple, nous allons utiliser la table suivante :

Documents	
Numéro	**L**
Sujet	**A**
Thème	**A**
Description	**T**
Marqueur création	**L**
Marqueur modification	**L**

Plutôt que de sauvegarder une date et une heure dans chaque enregistrement, vous pouvez stocker un "marqueur" dont la valeur exprime le nombre de secondes écoulées depuis une date antérieure arbitraire (dans cet exemple, le 1er janvier 1995 à 00:00:00) ainsi que la date et l'heure de la sauvegarde de l'enregistrement.

Dans notre exemple, le champ [Documents]Marqueur création contient le marqueur de création de l'enregistrement et le champ [Documents]Marqueur modification contient le marqueur de la dernière modification de l'enregistrement.

La méthode projet marqueurTemps suivante calcule le marqueur de temps par rapport à une date et une heure spécifiques ou par rapport à la date et l'heure courantes si aucun paramètre n'est passé :

```
` Méthode projet marqueurTemps
` marqueurTemps { ( Date ; Heure ) } -> Entier long
` marqueurTemps { ( Date ; Heure ) } -> Nombre de secondes depuis le 01/01/1995

C_DATE($1;$vdDate)
C_HEURE($2;$vhTime)
C_ENTIER LONG($0)

Si (Nombre de parametres=0)
   $vdDate:=Date du jour
   $vhTime:=Heure courante
Sinon
   $vdDate:=$1
   $vhTime:=$2
Fin de si
$0:=(($vdDate-!01/01/95!)*86400)+$vhTime
```

Note : Avec cette méthode, vous pouvez encoder toutes les dates et les heures situées entre le 01/01/95 à 00:00:00 et le 19/01/2063 à 03:14:07, ce qui représente l'intervalle de données exploitables par un entier long (de 0 à 2^31 moins 1).

A l'inverse, les méthodes projet Marqueur vers date et Marqueur vers heure vous permettent d'extraire la date et l'heure stockées dans un marqueur :

```
` Méthode projet Marqueur vers date
` Marqueur vers date ( Entier long ) -> Date
` Marqueur vers date ( Marqueur ) -> Date extraite

C_DATE($0)
C_ENTIER LONG($1)

$0:=!01/01/95!+($1\86400)
```

```
` Méthode projet Marqueur vers heure
` Marqueur vers heure ( Entier long ) -> Heure
` Marqueur vers heure ( Marqueur ) -> Heure extraite

C_HEURE($0)
C_ENTIER LONG($1)

$0:=Heure(Chaine heure(†00:00:00†+($1%86400)))
```

Pour vous assurer que les marqueurs des enregistrements sont correctement mis à jour, quelle que soit la manière dont ils sont créés ou modifiés, il suffit de faire appliquer cette règle par le trigger de la table [Documents]:

```
` Trigger de la table [Documents]

Au cas ou
   : (Evenement moteur=Sauvegarde nouvel enreg)
      [Documents]Marqueur création:=marqueurTemps
      [Documents]Marqueur modification:=marqueurTemps
   : (Evenement moteur=Sauvegarde enregistrement)
      [Documents]Marqueur modification:=marqueurTemps
Fin de cas
```

Une fois que cela est implémenté dans votre base, il suffit d'écrire la méthode projet CREER DOCUMENTATION listée ci-dessous. Nous utilisons PROPRIETES DOCUMENT et CHANGER PROPRIETES DOCUMENT pour gérer la date et l'heure de création et de modification des documents.

```
   ` Méthode projet CREER DOCUMENTATION

C_ALPHA(255;$vsPath;$vsDocPathName;$vsDocName)
C_ENTIER LONG($vlDoc)
C_BOOLEEN($vbOnWindows;$vbDoIt;$vbLocked;$vbInvisible)
C_HEURE($vhDocRef;$vhCreatedAt;$vhModifiedAt)
C_DATE($vdCreatedOn;$vdModifiedOn)

Si (Type application=4D Client)
      ` Si 4D Client est utilisé, sauvegarder les documents localement
      ` c'est-à-dire sur le poste client où se trouve 4D Client
   $vsPath:=Nom long vers chemin d'accès (Type application)
Sinon
      ` Sinon, sauvegarder les documents là où se trouve le fichier de données
   $vsPath:=Nom long vers chemin d'accès (Fichier donnees)
Fin de si
   ` Stocker les documents dans un répertoire nommé arbitrairement "Documentation"
$vsPath:=$vsPath+"Documentation"+Caractere(Symbole séparateur)
   ` Si ce répertoire n'existe pas, le créer
Si (Tester chemin acces ($vsPath) # Est un répertoire)
   CREER DOSSIER($vsPath)
Fin de si
   ` Etablir la liste des documents existants
   ` car nous allons devoir supprimer ceux qui sont obsolètes, autrement dit
   ` ceux dont les enregistrements correspondants ont été supprimés.
TABLEAU ALPHA(255;$asDocument;0)
LISTE DES DOCUMENTS($vsPath;$asDocument)
   ` Sélection de tous les enregistrements de la table [Documents]
TOUT SELECTIONNER([Documents])
   ` Pour chaque enregistremnt
$vlNbRecords:=Enregistrements trouves([Documents])
$vlNbDocs:=0
$vbOnWindows:=Sous Windows
Boucle ($vlDoc;1;$vlNbRecords)
      ` Supposons que nous aurons à (re)créer le document sur disque
   $vbDoIt:=Vrai
      ` Calcul du nom et du chemin d'accès au document
   $vsDocName:="DOC"+Chaine([Documents]Numéro;"00000")
   $vsDocPathName:=$vsPath+$vsDocName
```

```
      ` Est-ce que ce document existe déjà ?
   Si (Tester chemin acces($vsDocPathName+".HTM")=Est un document)
         ` Si oui, retirer le document de la liste des documents
         ` qui peuvent être supprimés
      $vlElem:=Chercher dans tableau($asDocument;$vsDocName+".HTM")
      Si ($vlElem>0)
         SUPPRIMER LIGNES($asDocument;$vlElem)
      Fin de si
         ` Est-ce que le document a été stocké après la dernière modification de
         ` l'enregistrement?
⇒     PROPRIETES DOCUMENT($vsDocPathName+".HTM";$vbLocked;$vbInvisible;
                          $vdCreatedOn;$vhCreatedAt;$vdModifiedOn;$vhModifiedAt)
      Si (marqueurTemps ($vdModifiedOn;$vhModifiedAt)>=[Documents]Marqueur
                                                              modification)
            ` Si oui, nous n'avons pas besoin de recréer le document
         $vbDoIt:=Faux
      Fin de si
   Sinon
         ` Le document n'existe pas, mettre ces deux variables à zéro, pour que
         ` nous sachions que nous devrons les traiter avant de fixer les propriétés
         ` finales du document
      $vdModifiedOn:=!00/00/00!
      $vhModifiedAt:=†00:00:00†
   Fin de si
      ` Avons-nous besoin de (re)créer le document?
   Si ($vbDoIt)
         ` Si oui, incrémenter le nombre de documents mis à jour
      $vlNbDocs:=$vlNbDocs+1
         ` Supprimer le document s'il existe déjà
      SUPPRIMER DOCUMENT($vsDocPathName+".HTM")
         ` Et le recréer
      Si ($vbOnWindows)
         $vhDocRef:=Creer document($vsDocPathName;"HTM")
      Sinon
         $vhDocRef:=Creer document($vsDocPathName+".HTM")
      Fin de si
      Si (OK=1)
            ` ...
            ` Ecrivons ici le contenu du document
            ` ...
         FERMER DOCUMENT($vhDocRef)
```

```
            Si ($vdModifiedOn=!00/00/00!)
                  ` Le document n'existait pas, fixer les valeurs correctes pour
                  ` la date et l'heure de modification
               $vdModifiedOn:=Date du jour
               $vhModifiedAt:=Heure courante
            Fin de si
               ` Changer les propriétés du document de telle manière que sa date et son
               ` heure de création soit égales à celles de l'enregistrement correspondant
            CHANGER PROPRIETES DOCUMENT($vsDocPathName+".HTM";$vbLocked;
                                                $vbInvisible;Marqueur vers date
                                              ([Documents]Marqueur création);
                                            Marqueur vers heure([Documents]Marqueur
                                              création);$vdModifiedOn;$vhModifiedAt)
         Fin de si
      Fin de si
         ` Juste pour savoir ce qui se passe
      CHANGER TITRE FENETRE("Traitement du document "+Chaine($vlDoc)+" sur "
                                                      +Chaine($vlNbRecords))
      ENREGISTREMENT SUIVANT([Documents])
   Fin de boucle
      ` Suppression des documents obsolètes, c'est-à-dire ceux
      ` qui sont toujours dans le tableau $asDocument
   Boucle ($vlDoc;1;Taille tableau($asDocument))
      SUPPRIMER DOCUMENT($vsPath+$asDocument{$vlDoc})
      CHANGER TITRE FENETRE("Suppression du document obsolète: "+Caractere(34)+
                                                $asDocument{$vlDoc}+Caractere(34))
   Fin de boucle
      ` C'est la fin
   ALERTE("Nombre de documents traités : "+Chaine($vlNbRecords)+Caractere(13)+
            "Nombre de documents mis à jour : "+Chaine($vlNbDocs)+Caractere(13)+
         "Nombre de documents supprimés : "+Chaine(Taille tableau($asDocument)))
```

Référence

CHANGER PROPRIETES DOCUMENT, Createur document, Type document.

CHANGER PROPRIETES DOCUMENT (document; verrouillé; invisible; créé le; créé à; modifié le; modifié à)

Paramètre	Type		Description
document	Alpha	→	Nom du document ou Chemin d'accès complet au document
verrouillé	Booléen	→	Verrouillé (Vrai) ou non verrouillé (Faux)
invisible	Booléen	→	Invisible (Vrai) ou visible (Faux)
créé le	Date	→	Date de création
créé à	Heure	→	Heure de création
modifié le	Date	→	Date de dernière modification
modifié à	Heure	→	Heure de dernière modification

Description

La commande CHANGER PROPRIETES DOCUMENT modifie certaines informations du document dont vous avez passé le nom ou le chemin d'accès dans document.

Avant l'appel :
• Passez Vrai dans verrouillé pour verrouiller le document. Un document verrouillé ne peut être ouvert ni supprimé. Passez Faux dans verrouillé pour déverrouiller un document.
• Passez Vrai dans invisible pour cacher le document. Passez Faux dans invisible pour rendre le document visible dans les fenêtres du bureau.
• Passez la date et l'heure de création du document dans créé le et créé à.
• Passez la date et l'heure de la dernière modification du document dans modifié le et modifié à.

L'heure et la date de création et de dernière modification sont gérées par le gestionnaire de fichiers de votre système, à chaque fois que vous créez ou modifiez un document. Cette commande vous permet de modifier ces propriétés, dans des buts particuliers. Reportez-vous à l'exemple de la commande PROPRIETES DOCUMENT.

Référence

CHANGER CREATEUR DOCUMENT, CHANGER TYPE DOCUMENT, PROPRIETES DOCUMENT.

LIRE ICONE DOCUMENT (cheminDoc; icône{; taille})

Paramètre	Type		Description
cheminDoc	Alpha	→	Nom ou chemin d'accès du fichier duquel obtenir l'icône ou chaîne vide pour afficher la boîte de dialogue d'ouverture de fichier
icône	Image	→	Champ ou variable image
		←	Icône du document
taille	Entier long	→	Taille de l'icône (en pixels)

Description

La commande LIRE ICONE DOCUMENT retourne dans le champ ou la variable image 4D icône, l'icône du fichier dont vous avez passé le nom ou le chemin d'accès complet dans cheminDoc. Le fichier peut être de tout type (document, exécutable, raccourci ou alias...). Cependant, la commande ne retourne pas les icônes de dossiers.

Passez dans cheminDoc le chemin d'accès absolu du fichier dont vous souhaitez récupérer l'icône. Vous pouvez passer uniquement le nom du fichier ou un chemin d'accès relatif, dans ce cas le fichier doit se trouver dans le dossier courant de la base (généralement, le dossier contenant le fichier de structure de la base).
Si vous passez une chaîne vide dans cheminDoc, la boîte de dialogue standard d'ouverture de fichiers apparaît, permettant à l'utilisateur de désigner un fichier. Une fois la boîte de dialogue validée, la variable système *Document* contient le chemin d'accès complet du document sélectionné.

Passez dans le paramètre icône un champ ou une variable image 4D. Après l'exécution de la commande, ce paramètre contient l'icône du fichier (au format PICT).

Le paramètre optionnel taille vous permet d'indiquer les dimensions de l'image que vous souhaitez obtenir. La valeur du paramètre correspond à la longueur d'un côté du carré dans lequel l'image sera incluse. Généralement, les icônes sont définies en 32x32 pixels ("grande icône") ou 16x16 pixels ("petite icône"). Si vous passez 0 ou omettez le paramètre, la commande retourne l'icône dans sa plus grande taille disponible.

Taille document

Documents système

version 6.0

Taille document (document{; *}) → Numérique

Paramètre	Type		Description
document	DocRef | Alpha	→	Numéro de référence de document ou Nom de document
*		→	MacOS uniquement : - si omis, taille de la data fork - si passé, taille de la resource fork
Résultat	Numérique	←	Taille (en octets) de document

Description

La commande Taille document retourne la taille, exprimée en octets, d'un document.

Si le document est ouvert, passez son numéro de référence dans document.
Si le document n'est pas ouvert, passez son nom ou son chemin d'accès dans document.

Sous MacOS, si vous ne passez pas le paramètre optionnel *, la taille de la data fork est retournée. Si vous passez le paramètre optionnel *, la taille de la resource fork est retournée.

Référence

CHANGER POSITION DANS DOCUMENT, CHANGER TAILLE DOCUMENT, Position dans document.

CHANGER TAILLE DOCUMENT (document; taille)

Paramètre	Type		Description
document	DocRef	→	Numéro de référence de document
taille	Numérique	→	Nouvelle taille (en octets) de document

Description

La commande CHANGER TAILLE DOCUMENT fixe la taille d'un document au nombre d'octets que vous avez passé dans taille.

Le document doit avoir été ouvert au préalable. Vous passez son numéro de référence dans document.

Sous MacOS, c'est la taille de la data fork du document qui est modifiée.

Référence

CHANGER POSITION DANS DOCUMENT, Position dans document, Taille document.

Position dans document

Documents système

version 6.0

Position dans document (docRef) → Numérique

Paramètre	Type		Description
docRef	DocRef	→	Numéro de référence de document
Résultat	Numérique	←	Position dans le fichier (exprimée en octets) à partir du début du fichier

Description

Cette commande ne fonctionne qu'avec un document déjà ouvert, dont vous avez passé le numéro de référence dans le paramètre document.

Position dans document retourne la position, à partir du début du document, à laquelle la prochaine lecture (RECEVOIR PAQUET) ou écriture (ENVOYER PAQUET) aura lieu.

Référence

CHANGER POSITION DANS DOCUMENT, ENVOYER PAQUET, RECEVOIR PAQUET.

CHANGER POSITION DANS DOCUMENT

Documents système

version 6.0

CHANGER POSITION DANS DOCUMENT (docRef; offset{; ancre})

Paramètre	Type		Description
docRef	RefDoc	→	Numéro de référence de document
offset	Numérique	→	Position dans fichier (exprimée en octets)
ancre		→	1 = Par rapport au début du fichier 2 = Par rapport à la fin du fichier 3 = Par rapport à la position courante

Description

Cette commande ne fonctionne qu'avec un document déjà ouvert, dont vous avez passé le numéro de référence dans le paramètre docRef.

CHANGER POSITION DANS DOCUMENT définit la position que vous passez dans offset comme étant celle à laquelle la prochaine lecture (RECEVOIR PAQUET) ou écriture (ENVOYER PAQUET) aura lieu.

Si vous omettez le paramètre optionnel ancre, la position est définie par rapport au début du document. Sinon, vous pouvez passer dans le paramètre ancre une des valeurs listées ci-dessus.

En fonction de l'ancre définie, vous pouvez passer des valeurs positives ou négatives dans le paramètre offset.

Référence

ENVOYER PAQUET, Position dans document, RECEVOIR PAQUET.

13

Enregistrements

AFFICHER ENREGISTREMENT

Enregistrements

version 5

AFFICHER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle afficher l'enregistrement courant ou Table par défaut si ce paramètre est omis

Description

AFFICHER ENREGISTREMENT affiche l'enregistrement courant de table dans le formulaire entrée courant. L'enregistrement reste affiché jusqu'à ce qu'un événement provoque un redessinement de la fenêtre. Cet événement peut être l'exécution d'un AJOUTER ENREGISTREMENT, le retour au formulaire entrée ou à la barre de menus. AFFICHER ENREGISTREMENT ne fait rien s'il n'y a pas d'enregistrement courant.

AFFICHER ENREGISTREMENT est souvent utilisé pour afficher des messages de progression personnalisés. Cette commande peut également servir à générer un "slide show" automatique.

Si une méthode formulaire existe, un événement Sur chargement est généré.

Exemple

L'exemple suivant affiche une série d'enregistrements sous forme de slide show :

```
   TOUT SELECTIONNER([Démo]) ` Sélection de tous les enregistrements
   FORMULAIRE ENTREE ([Démo]; "Affichage") ` Désignation du formulaire à utiliser
   Boucle ($i; 1; Enregistrements trouves([Démo])) ` Boucle sur tous les enregistrements
⇒     AFFICHER ENREGISTREMENT([Démo]) ` Afficher un enregistrement
      ENDORMIR PROCESS (Numero du process courant; 180) ` 3 secondes de pause
      ENREGISTREMENT SUIVANT([Démo]) ` Passage à l'enregistrement suivant
   Fin de boucle
```

Référence

MESSAGE.

CREER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table dans laquelle créer un enregistrement ou Table par défaut si ce paramètre est omis

Description

CREER ENREGISTREMENT crée un nouvel enregistrement vide pour la table table, mais ne l'affiche pas à l'écran. Vous devez utiliser la commande AJOUTER ENREGISTREMENT pour créer un nouvel enregistrement et l'afficher dans un formulaire entrée.

Utilisez CREER ENREGISTREMENT plutôt que AJOUTER ENREGISTREMENT lorsque les valeurs de l'enregistrement sont entrées par programmation. Le nouvel enregistrement devient à la fois l'enregistrement courant et la sélection courante (la sélection courante ne contient qu'un enregistrement).

L'enregistrement est créé uniquement en mémoire et doit être sauvegardé à l'aide de STOCKER ENREGISTREMENT. Si vous changez d'enregistrement courant (par exemple à la suite d'une recherche) avant la sauvegarde, l'enregistrement créé est perdu.

Exemple

Cet exemple archive les enregistrements datant de plus de 30 jours. L'opération est réalisée par la création d'enregistrements dans une table d'archive. Une fois l'opération terminée, les enregistrements archivés sont supprimés de la table [Comptes] :

```
    ` Recherche des enregistrements datant de plus de 30 jours
CHERCHER ([Comptes]; [Comptes]Saisie < (Date du jour – 30))
Boucle ($vlRecord; 1; Enregistrements trouves([Comptes]))
⇒    CREER ENREGISTREMENT ([Archives]) ` Création d'un nouvel enregistrement
     [Archive]Numéro := [Comptes]Number ` Copie des champs dans l'archive
     [Archive]Saisie := [Comptes]Saisie
     [Archive]Montant := [Comptes]Montant
     STOCKER ENREGISTREMENT ([Archive]) ` Sauvegarde de l'enregistrement d'archive
     ENREGISTREMENT SUIVANT ([Comptes]) ` Passage à l'enregistrement suivant
Fin de boucle
SUPPRIMER SELECTION([Comptes]) ` Suppression des enregistrements archivés
```

Référence

STOCKER ENREGISTREMENT.

DUPLIQUER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table de l'enregistrement à dupliquer ou Table par défaut si ce paramètre est omis

Description

DUPLIQUER ENREGISTREMENT duplique l'enregistrement courant de table. Ce nouvel enregistrement devient l'enregistrement courant. S'il n'y a pas d'enregistrement courant, DUPLIQUER ENREGISTREMENT ne fait rien. Appelez la commande STOCKER ENREGISTREMENT pour sauvegarder le nouvel enregistrement.

DUPLIQUER ENREGISTREMENT peut être exécuté pendant la saisie des données. Vous pouvez donc dupliquer l'enregistrement qui est affiché à l'écran. N'oubliez pas que vous devez d'abord appeler STOCKER ENREGISTREMENT si vous voulez sauvegarder les modifications apportées à l'enregistrement original.

Référence

STOCKER ENREGISTREMENT.

Nouvel enregistrement {(table)} → Booléen

Paramètre	Type		Description
table	Table	→	Table de l'enregistrement à examiner ou Table par défaut si ce paramètre est omis
Résultat	Booléen	←	Vrai si l'enregistrement est en cours de création, Faux sinon

Description

La commande Nouvel enregistrement retourne Vrai lorsque l'enregistrement courant de table est en cours de création et n'a pas encore été sauvegardé dans le process courant.

Note de compatibilité : Il est possible d'obtenir la même information avec la commande existante Numero enregistrement, en testant si elle retourne -3. Toutefois, il est vivement conseillé d'utiliser dans ce cas Nouvel enregistrement plutôt que Numero enregistrement. En effet, la commande Nouvel enregistrement assure une meilleure compatibilité avec les futures versions de 4e Dimension.

Exemple

Les deux instructions suivantes sont identiques, la seconde est conseillée pour que le code reste compatible avec les prochaines versions de 4D :

```
Si (Numero enregistrement([Table])=-3) `Déconseillé
   ` ...
Fin de si

⇒ Si (Nouvel enregistrement([Table])) `Conseillé
   ` ...
Fin de si
```

Référence

Enregistrement modifie, Numero enregistrement.

Enregistrement modifie {(table)} → Booléen

Paramètre	Type		Description
table	Table	→	Table de laquelle tester si l'enregistrement courant a été modifié ou Table par défaut si ce paramètre est omis
Résultat	Booléen	←	L'enregistrement a été modifié (Vrai) ou L'enregistrement n'a pas été modifié (Faux)

Description

Enregistrement modifie retourne Vrai si l'enregistrement courant de table a été modifié et non encore stocké. Sinon, elle retourne Faux. Cette fonction vous permet de déterminer rapidement s'il faut stocker l'enregistrement. Dans les formulaires entrée, vous pouvez effectuer le test avant d'aller à l'enregistrement suivant. Cette fonction retourne toujours Vrai pour un nouvel enregistrement.

Exemples

L'exemple suivant montre une utilisation typique de Enregistrement modifie :

```
⇒ Si (Enregistrement modifie ([Clients]))
     STOCKER ENREGISTREMENT ([Clients])
  Fin de si
```

Référence

Ancien, Modifie, STOCKER ENREGISTREMENT.

Enregistrement charge {(table)} → Booléen

Paramètre	Type		Description
table	Table	→	Table de l'enregistrement à examiner ou Table par défaut si ce paramètre est omis
Résultat	Booléen	←	Vrai si l'enregistrement est chargé, Faux sinon

Description

La commande Enregistrement charge retourne Vrai si l'enregistrement courant de table est chargé dans le process en cours.

Exemple

Au lieu d'utiliser les actions automatiques "Enregistrement suivant" ou "Enregistrement précédent", vous voulez écrire dans les méthodes de boutons sans action des instructions spécifiant que le bouton "Suivant" affiche le début de la sélection si la fin de la sélection est atteinte et que le bouton "Précédent" affiche la fin de la sélection si le début est atteint.

```
      ` Méthode objet du bouton sans action "PRÉCÉDENT"
   Si (Evenement formulaire=Sur clic souris)
      ENREGISTREMENT PRECEDENT([Groupe])
⇒     Si (Non(Enregistrement charge([Groupe])))
         ALLER DANS SELECTION([Groupe];Enregistrements trouves([Groupe]))
               `Aller au dernier enregistrement de la sélection
      Fin de si
   Fin de si

      ` Méthode objet du bouton sans action "SUIVANT"
   Si (Evenement formulaire=Sur clic souris)
      ENREGISTREMENT SUIVANT([Groupe])
⇒     Si (Non(Enregistrement charge([Groupe])))
         ALLER DANS SELECTION([Groupe];1)
               `Aller au premier enregistrement de la sélection
      Fin de si
   Fin de si
```

STOCKER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table de l'enregistrement à stocker ou Table par défaut si ce paramètre est omis

Description

STOCKER ENREGISTREMENT sauvegarde l'enregistrement courant de table pour le process courant. S'il n'y a pas d'enregistrement courant, la commande est ignorée.

Vous pouvez utiliser STOCKER ENREGISTREMENT pour sauvegarder un enregistrement créé ou modifié par programmation. Lorsqu'un enregistrement a été modifié puis validé par un utilisateur dans un formulaire, il n'est pas nécessaire de le sauvegarder à l'aide de STOCKER ENREGISTREMENT. En revanche, un enregistrement modifié puis annulé par l'utilisateur peut malgré tout être sauvegardé avec STOCKER ENREGISTREMENT.

L'utilisation de STOCKER ENREGISTREMENT est nécessaire dans les cas suivants :
• Pour sauvegarder un enregistrement créé par les commandes CREER ENREGISTREMENT ou DUPLIQUER ENREGISTREMENT,
• Pour sauvegarder des données issues de la commande RECEVOIR ENREGISTREMENT,
• Pour sauvegarder un enregistrement modifié par une méthode,
• Pour sauvegarder un enregistrement contenant un sous-enregistrement ayant été créé ou modifié par la commande AJOUTER SOUS ENREGISTREMENT, CREER SOUS ENREGISTREMENT ou MODIFIER SOUS ENREGISTREMENT,
• Pendant la saisie de données, pour sauvegarder l'enregistrement affiché avant d'appeler une commande qui change l'enregistrement courant,
• Pendant la saisie de données, pour sauvegarder l'enregistrement courant.

Vous ne devez pas appeler STOCKER ENREGISTREMENT dans l'événement formulaire Sur validation d'un enregistrement qui a été validé, sinon l'enregistrement est sauvegardé deux fois.

Exemple

L'exemple suivant est une partie d'une méthode récupérant des enregistrements d'un fichier. Dans cette partie, les enregistrements sont reçus puis, si l'opération s'est correctement déroulée, sauvegardés :

```
      ` Réception de l'enregistrement à partir du disque
   RECEVOIR ENREGISTREMENT ([Clients])
   Si (OK = 1)   ` Si l'enregistrement a été correctement reçu...
⇒     STOCKER ENREGISTREMENT ([Clients])   ` Le sauvegarder
   Fin de si
```

Référence

CREER ENREGISTREMENT, Enregistrement verrouille, Présentation des triggers.

SUPPRIMER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle supprimer l'enregistrement courant ou Table par défaut si ce paramètre est omis

Description

SUPPRIMER ENREGISTREMENT supprime de table l'enregistrement courant du process en cours. S'il n'y a pas d'enregistrement courant pour table dans le process, SUPPRIMER ENREGISTREMENT ne fait rien. Dans un formulaire, vous pouvez créer un bouton 'Supprimer enregistrement' et lui assigner l'action automatique correspondante, plutôt que d'utiliser cette commande. Une fois que l'enregistrement est supprimé, la sélection courante de table est vide.

La suppression d'enregistrements est une opération définitive et ne peut être annulée.

Lorsqu'un enregistrement est supprimé, son numéro interne est réutilisé lors de la création de nouveaux enregistrements. Par conséquent, n'utilisez pas ces numéros comme identifiants de vos enregistrements si votre base permet la suppression d'enregistrements.

Exemple

L'exemple suivant permet de supprimer l'enregistrement d'un employé. La méthode demande à l'utilisateur le numéro de l'employé à supprimer, recherche l'enregistrement correspondant puis le supprime :

```
      ` On récupère un numéro d'identification
   vCherch := Demander ("Numéro de l'employé à supprimer :")
   Si (OK = 1)
      CHERCHER ([Employés]; [Employés]Numéro = vCherch)   ` Trouver l'employé
⇒     SUPPRIMER ENREGISTREMENT ([Employés])   ` Suppression de l'enregistrement
   Fin de si
```

Référence

Enregistrement verrouille, Présentation des triggers.

Enregistrements dans table

Enregistrements dans table {(table)} → Numérique

Paramètre	Type		Description
table	Table	→	Table de laquelle retourner le nombre total d'enregistrements ou Table par défaut si ce paramètre est omis
Résultat	Numérique	←	Nombre total d'enregistrements dans table

Description

Enregistrements dans table retourne le nombre total d'enregistrements que contient table. Par opposition, Enregistrements trouves retourne le nombre d'enregistrements de la sélection courante uniquement. Lorsque cette commande est utilisée dans une transaction, les enregistrements éventuellement créés pendant la transaction sont comptabilisés.

Exemple

L'exemple suivant affiche une alerte indiquant le nombre d'enregistrements de la table :

```
⇒ ALERTE ("Il y a " + Chaine (Enregistrements dans table ([Personnes])) +
                                                    " enregistrements dans la table.")
```

Référence

Enregistrements trouves.

Numero enregistrement {(table)} → Numérique

Paramètre	Type		Description
table	Table	→	Table de laquelle vous souhaitez obtenir le numéro de l'enregistrement courant ou Table par défaut si ce paramètre est omis
Résultat	Numérique	←	Numéro d'enregistrement courant

Description

Numero enregistrement retourne le numéro de l'enregistrement courant de table. S'il n'y a pas d'enregistrement courant, par exemple si le pointeur d'enregistrement se trouve avant ou après la sélection courante, Numero enregistrement retourne –1. S'il s'agit d'un nouvel enregistrement qui n'a pas encore été sauvegardé, Numero enregistrement retourne –3.

Les numéros d'enregistrements peuvent varier. Par exemple, les numéros des enregistrements supprimés sont réutilisés. Les numéros d'enregistrements changent aussi si vous compactez votre base ou si vous faites une réparation par analyse des marqueurs d'enregistrements à l'aide de 4D Util. Pendant une transaction, un enregistrement nouvellement créé a un numéro d'enregistrement temporaire. Après la validation de la transaction, l'enregistrement recevra un numéro d'enregistrement standard.

Exemple

L'exemple suivant sauvegarde le numéro d'enregistrement courant puis cherche dans la table si un autre enregistrement a la même valeur :

```
⇒   $NumEnreg := Numero enregistrement ([Personnes])  ` Obtenir le n° d'enregistrement
      ` Est-ce que quelqu'un d'autre a le même nom ?
    CHERCHER ([Personnes]; [Personnes]Nom = [Personnes]Nom)
      ` Afficher une alerte avec le nombre de personnes qui ont le même nom
    ALERTE ("Il existe " + Chaine (Enregistrements trouves ([Personnes])
                                                  + " personnes du même nom.")
    ALLER A ENREGISTREMENT ([Personnes]; $NumEnreg)
      ` Retour à l'enregistrement original
```

Référence

A propos des numéros d'enregistrements, ALLER A ENREGISTREMENT, Nouvel enregistrement, Numero dans selection, Numerotation automatique.

ALLER A ENREGISTREMENT ({table; }enregistrement)

Paramètre	Type		Description
table	Table	→	Table de l'enregistrement de destination ou Table par défaut si ce paramètre est omis
enregistrement	Numérique	→	Numéro renvoyé par Numero enregistrement

Description

ALLER A ENREGISTREMENT sélectionne l'enregistrement courant de table. Le paramètre enregistrement est le numéro renvoyé par la fonction Numero enregistrement. Après l'exécution de cette commande, l'enregistrement est le seul de la sélection courante.

Si enregistrement est inférieur au plus petit numéro d'enregistrement ou supérieur au plus grand numéro d'enregistrement de la base, 4e Dimension génère un message d'erreur indiquant que le numéro est hors intervalle. Si enregistrement est égal au numéro d'un enregistrement supprimé, la sélection courante devient vide.

Note : Vous ne devez pas utiliser avec cette commande les numéros d'enregistrements temporaires renvoyés pendant une transaction.

Exemple

Référez-vous à l'exemple de Numero enregistrement.

Référence

A propos des numéros d'enregistrements, Numero enregistrement.

Numerotation automatique {(table)} → Numérique

Paramètre	Type		Description
table	Table	→	Table à numéroter automatiquement ou Table par défaut si ce paramètre est omis
Résultat	Numérique	←	Numéro automatique

Description

Numerotation automatique retourne le prochain numéro automatique de table. Ce numéro est unique pour chaque table. C'est une valeur qui ne se répète pas et qui est incrémentée à chaque enregistrement nouvellement créé dans la table. La numérotation commence à 1.

Le numéro retourné par cette fonction est identique à celui retourné lorsque vous fixez #N comme valeur par défaut pour un champ dans un formulaire (référez-vous au manuel *Mode Structure* de 4e Dimension pour plus d'informations sur l'affectation des valeurs par défaut).

Dans les cas suivants, il est indispensable d'utiliser la fonction Numerotation automatique plutôt que #N :
• Si vous créez des enregistrements par l'intermédiaire d'une méthode
• Si la numérotation ne doit pas commencer à 1
• Si la numérotation doit s'incrémenter d'un nombre supérieur à 1
• Si le numéro doit reprendre une partie d'un code

Pour stocker ce numéro à l'aide d'une méthode, il faut créer un champ de type Entier long dans la table et y affecter la numérotation automatique.

Si la numérotation doit débuter à une valeur différente de 1, ajoutez simplement la différence à la fonction Numerotation automatique. Par exemple, si le numéro doit commencer à 1000, vous pouvez utiliser la ligne de code suivante pour affecter le numéro :

```
⇒ [Table1]NumAuto := Numerotation automatique ([Table1]) + 999
```

Exemple

L'exemple suivant fait partie d'une méthode formulaire. Ces lignes de code testent s'il s'agit d'un nouvel enregistrement (si le numéro de facture est égal à une chaîne vide). Si l'enregistrement est nouveau, un numéro est affecté à la facture. Ce numéro de facture est construit avec deux informations : le numéro unique et le numéro de référence de l'utilisateur, qui était saisi à l'ouverture de la base. Le numéro unique est formaté en tant que chaîne avec une longueur de cinq caractères :

```
            ` S'il s'agit d'une nouvelle facture, créer un numéro de facture
      Si ([Factures]NumFacture = "")
            ` Le numéro de facture est une chaîne qui se termine par le numéro de référence
            ` de l'utilisateur
⇒        [Factures]NumFacture := Chaine (Numerotation automatique; "00000")
                                                              + [Factures]Utilisateur
      Fin de si
```

Référence

A propos des numéros d'enregistrements, Numero dans selection, Numero enregistrement.

A propos des numéros d'enregistrements

Dans 4e Dimension, trois numéros sont associés à un enregistrement :
• Numéro d'enregistrement,
• Numéro dans la sélection,
• Numéro automatique.

Numéro d'enregistrement

Le numéro d'enregistrement est le numéro physique/absolu de l'enregistrement. Ce numéro est automatiquement assigné à chaque nouvel enregistrement et reste le même jusqu'à ce que cet enregistrement soit détruit ou que la table soit retriée de façon permanente avec 4D Util. Les enregistrements commencent au numéro zéro (0).
Les numéros d'enregistrements ne sont pas uniques car les numéros des enregistrements détruits sont réutilisés pour de nouveaux enregistrements. Ces numéros sont également modifiés lors du tri permanent de la table (avec 4D Util) et lorsque la base est réparée ou compactée. Les nouveaux enregistrements ajoutés dans une transaction reçoivent des numéros temporaires (incrémentés à partir du numéro 18 000 000) ; ils ne reçoivent des numéros définitifs que lorsque la transaction est validée.

Numéro dans la sélection

Le numéro dans la sélection est la position de l'enregistrement dans la sélection courante. Ce numéro dépend de la sélection courante. Si la sélection est modifiée ou triée, ce numéro change aussi probablement. La numérotation dans une sélection courante commence à un (1).

Numéro automatique

Le numéro automatique est un numéro unique, non répétable, qui peut être assigné à un champ dans un enregistrement. Il n'est pas automatiquement stocké à chaque enregistrement. Il démarre à 1 et est incrémenté à chaque création d'un nouvel enregistrement. A la différence des numéros d'enregistrements, un numéro automatique n'est pas réutilisé lorsque l'enregistrement est détruit, ou lorsque la base est compactée, réparée, ou retriée de façon permanente par 4D Util.
Ces numéros fournissent un moyen d'attribuer un numéro d'identification unique à chaque enregistrement. Si un numéro automatique est incrémenté pendant une transaction, ce numéro n'est pas décrémenté si la transaction est annulée

Exemples de numéros d'enregistrements

Les tableaux suivants comparent le fonctionnement des différents numéros d'enregistrements. Chaque ligne de tableau représente les informations d'un enregistrement. L'ordre des lignes est celui dans lequel les enregistrements seraient affichés dans un formulaire sortie.

• **Colonne des Données** : Les valeurs d'un champ dans chaque enregistrement. Elle contient le nom d'une personne.
• **Colonne de Numéro d'enregistrement (N° Enrg)** : C'est le numéro absolu de l'enregistrement et qui est retourné par la fonction Numero enregistrement.
• **Colonne de Numéro dans la sélection (N° Sélection)** : C'est le numéro de position dans la sélection courante, qui est retourné par la fonction Numero dans selection.
• **Colonne de Numéro automatique (N° Auto)**: C'est le numéro unique de l'enregistrement, qui est retourné par la fonction Numerotation automatique. Ce numéro est stocké dans un champ.

Après saisie des enregistrements :
Le premier tableau présente des enregistrements qui viennent d'être saisis.
• l'ordre des enregistrements par défaut est le numéro d'enregistrement.
• Le numéro d'enregistrement commence à 0.
• Le numéro dans la sélection et le numéro automatique commencent à 1.

Données	N° Enrg	N° Sélection	N° Auto
Tess	0	1	1
Terri	1	2	2
Sabra	2	3	3
Sam	3	4	4
Lisa	4	5	5

Note: Les enregistrements restent dans l'ordre par défaut après l'appel de toute commande qui modifie la sélection sans la réordonner, comme par exemple la commande de menu **Tout montrer** en mode Utilisation ou après l'exécution de la commande TOUT SELECTIONNER.

Après un tri des enregistrements :
La première partie du tableau présente les enregistrements triés par noms.
• Le numéro d'enregistrement reste associé à l'enregistrement.
• Le numéro dans la sélection reflète la nouvelle position de l'enregistrement dans la sélection triée.
• Le numéro automatique ne change jamais puisqu'il est assigné à la création de chaque enregistrement et stocké avec lui.

Données	N° Enrg	N° Sélection	N° Auto
Lisa	4	1	5
Sabra	2	2	3
Sam	3	3	4
Terri	1	4	2
Tess	0	5	1

Après la suppression d'un enregistrement :
Voici le tableau après la destruction de l'enregistrement de Sam.
• Seuls les numéros dans la sélection ont changé (les numéros dans la sélection reflètent l'ordre d'affichage des enregistrements).

Données	N° Enrg	N° Sélection	N° Auto
Lisa	4	1	5
Sabra	2	2	3
Terri	1	3	2
Tess	0	4	1

Après l'ajout d'un enregistrement :
Voici le tableau après l'ajout de l'enregistrement Liz.
• Un nouvel enregistrement est ajouté à la fin de la sélection courante.
• Le numéro d'enregistrement de Sam est réutilisé pour le nouvel enregistrement.
• Le numéro automatique a été incrémenté de 1.

Données	N° Enrg	N° Sélection	N° Auto
Tess	0	1	1
Terri	1	2	2
Sabra	2	3	3
Lisa	4	4	5
Liz	3	5	6

Après un changement de sélection et un tri :
Le tableau qui suit montre les enregistrements après réduction de la sélection à trois enregistrements qui sont ensuite triés.
• Seuls le numéro dans la sélection change.

Données	N° Enrg	N° Sélection	N° Auto
Sabra	2	1	3
Liz	3	2	6
Terri	1	3	2

Référence
Numero dans selection, Numero enregistrement, Numerotation automatique.

EMPILER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle empiler l'enregistrement courant ou Table par défaut si ce paramètre est omis

Description

EMPILER ENREGISTREMENT "empile" une copie l'enregistrement courant de table (ainsi que ses sous-enregistrements, le cas échéant) dans la pile d'enregistrements de la table. EMPILER ENREGISTREMENT peut être exécuté avant qu'un enregistrement soit sauvegardé.

Si vous empilez un enregistrement non verrouillé, il sera verrouillé pour tous les autres process et utilisateurs jusqu'à ce que vous le "dépiliez" (c'est-à-dire que vous le déchargiez de la pile).

Exemple

L'exemple suivant empile l'enregistrement d'un client :

```
⇒ EMPILER ENREGISTREMENT ([Client]) ` Placer l'enregistrement du client dans la pile
```

Référence

DEPILER ENREGISTREMENT, Utiliser la pile d'enregistrements.

DEPILER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle dépiler l'enregistrement courant ou Table par défaut si ce paramètre est omis

Description

DEPILER ENREGISTREMENT charge le premier enregistrement de la pile d'enregistrements de table, et en fait l'enregistrement courant.

Si vous empilez un enregistrement puis créez une nouvelle sélection courante ne contenant plus l'enregistrement empilé, et enfin dépilez l'enregistrement, vous obtenez la situation dans laquelle l'enregistrement courant ne se trouve pas dans la sélection courante. Si vous souhaitez faire de l'enregistrement empilé la sélection courante, utilisez la commande ENREGISTREMENT SELECTION.
Si vous utilisez une routine qui déplace le pointeur d'enregistrement courant avant de sauvegarder l'enregistrement, vous perdrez la copie empilée en mémoire.

Exemples

L'exemple suivant récupère l'enregistrement d'un client dans la pile :

```
⇒ DEPILER ENREGISTREMENT ([Clients]) ` Dépiler l'enregistrement
```

Référence

EMPILER ENREGISTREMENT, Utiliser la pile d'enregistrements.

Les commandes EMPILER ENREGISTREMENT et DEPILER ENREGISTREMENT vous permettent de poser (“empiler”) des enregistrements sur le dessus de la pile des enregistrements, et de les enlever (“dépiler”) de la pile.

Chaque process dispose de sa propre pile d'enregistrements pour chaque table. 4D gère pour vous les piles d'enregistrements. Chaque pile d'enregistrements est du type LIFO ("Last-In-First-Out", ce qui peut se traduire par "dernier-entré-premier-sorti"). La capacité de la pile dépend de la mémoire.

Les commandes EMPILER ENREGISTREMENT et DEPILER ENREGISTREMENT doivent être utilisées avec prudence. Chaque enregistrement empilé utilise une partie de la mémoire disponible. Empiler trop d'enregistrements peut causer l'apparition d'un message du type "mémoire insuffisante" ou une pile pleine.

4D efface de la pile les enregistrements "dépilés" quand vous retournez au menu à la fin de l'exécution de la méthode.

EMPILER ENREGISTREMENT et DEPILER ENREGISTREMENT sont utiles lorsque par exemple, en cours de saisie, vous voulez examiner des enregistrements se trouvant dans la même table que celle que vous êtes en train d'utiliser. Pour cela, vous empilez votre enregistrement, cherchez et examinez les enregistrements dans la table (vous copiez des champs dans des variables, par exemple), et finalement vous dépilez l'enregistrement pour le restaurer.

Note pour les utilisateurs de la version 5 de 4D : Quand vous saisissez un enregistrement, si vous devez vérifier l'unicité d'une valeur sur plusieurs champs, utilisez la nouvelle commande FIXER DESTINATION RECHERCHE. Cela vous évitera les appels à EMPILER ENREGISTREMENT et DEPILER ENREGISTREMENT que vous deviez effectuer avant d'utiliser CHERCHER, afin de préserver les données saisies dans l'enregistrement courant. FIXER DESTINATION RECHERCHE permet d'exécuter une recherche qui ne change pas la sélection ni l'enregistrement courants.

Référence

DEPILER ENREGISTREMENT, EMPILER ENREGISTREMENT, FIXER DESTINATION RECHERCHE.

14

Enregistrements (verrouillage)

4e Dimension et 4D Server/4D Client gèrent automatiquement les bases en empêchant les conflits entre les process ou entre les utilisateurs. Deux utilisateurs ou deux process ne peuvent modifier en même temps le même enregistrement ou le même objet. Le second utilisateur ou process peut toutefois accéder simultanément en "lecture seulement" à l'enregistrement ou à l'objet.

Vous devez utiliser les commandes multi-utilisateurs de cette section dans plusieurs cas :
• Modification d'enregistrements par programmation,
• Utilisation d'une interface utilisateur personnalisée pour les opérations multi-utilisateurs,
• Sauvegarde de modifications reliées entre elles dans une transaction.

Il y a trois concepts importants à maîtriser lorsqu'on utilise des commandes dans une base multi-process :
1. Chaque table est soit en mode "lecture seulement" soit en mode "lecture/écriture".
2. Les enregistrements sont verrouillés quand ils sont chargés et déverrouillés quand ils sont libérés.
3. Un enregistrement verrouillé ne peut être modifié.

Dans les paragraphes suivants, la personne effectuant une opération dans la base multi-utilisateurs est l'utilisateur local. Les autres personnes utilisant la base sont appelées les autres utilisateurs. La discussion est menée du point de vue de l'utilisateur local. De même, du point de vue du multi-process, le process exécutant une opération dans la base est le process courant. Tout autre process en cours d'exécution est désigné comme un autre process. La discussion est menée du point de vue du process courant.

Enregistrements verrouillés

Un enregistrement verrouillé ne peut pas être modifié par l'utilisateur local ou le process courant. Un enregistrement verrouillé peut être chargé, mais pas modifié. Un enregistrement est verrouillé quand l'un des autres utilisateurs ou process a chargé l'enregistrement pour effectuer une modification. Seul l'utilisateur qui modifie l'enregistrement perçoit l'enregistrement comme étant non verrouillé. Tous les autres utilisateurs et process perçoivent l'enregistrement comme étant verrouillé, et donc indisponible pour modification. Une table doit être en mode lecture/écriture pour qu'un enregistrement puisse être chargé déverouillé (modifiable).

Etats Lecture seulement et Lecture/écriture

Chaque table d'une base est soit en mode lecture/écriture, soit en mode lecture seulement pour chaque utilisateur et process de la base. **Lecture seulement** signifie que les enregistrements de la table peuvent être chargés mais non modifiés. **Lecture/écriture** signifie que les enregistrements de la table peuvent être chargés et modifiés par un utilisateur/process, si aucun autre utilisateur/process n'a prélablement verrouillé l'enregistrement.

Notez que si vous changez l'état d'une table, celui-ci prend effet pour le prochain enregistrement chargé. Si un enregistrement est déjà chargé, la modification de l'état de la table ne l'affecte pas.

Etat lecture seulement

Lorsqu'une table est en lecture seulement et qu'un enregistrement est chargé, l'enregistrement est toujours verrouillé. En d'autres termes, l'enregistrement peut être affiché, imprimé et utilisé de diverses façons, mais ne peut être modifié.

Notez que le mode lecture seulement ne s'applique qu'à la modification d'enregistrements existants. Le mode lecture seulement n'affecte pas la création de nouveaux enregistrements : vous pouvez toujours ajouter des enregistrements à une table en lecture seulement en utilisant les commandes CREER ENREGISTREMENT et AJOUTER ENREGISTREMENT ou les commandes de menu du mode Utilisation.

4D met automatiquement une table en mode lecture seulement pour les commandes qui ne requièrent pas d'accès en écriture aux enregistrements. Ces commandes sont VISUALISER SELECTION, VALEURS DISTINCTES, ECRITURE DIF, ECRITURE SYLK, ECRITURE ASCII, GRAPHE SUR SELECTION, IMPRIMER SELECTION, IMPRIMER ETIQUETTES, ETAT, SELECTION VERS TABLEAU, SELECTION LIMITEE VERS TABLEAU.

Avant d'exécuter ces commandes, 4D sauvegarde l'état courant de la table (lecture seulement ou lecture/écriture) dans le process courant. Une fois la commande exécutée, l'état est restauré.

Etat lecture/écriture

Lorsqu'une table est en lecture/écriture et qu'un enregistrement est chargé, l'enregistrement sera non verrouillé si aucun autre utilisateur ne l'a préalablement chargé. Si l'enregistrement est verrouillé par un autre utilisateur, l'enregistrement est chargé verrouillé et ne peut être modifié par l'utilisateur local.

Une table doit être en mode lecture/écriture et l'enregistrement doit être chargé pour qu'il soit déverrouillé et donc modifiable.

Si un utilisateur charge un enregistrement d'une table en mode lecture/écriture, aucun autre utilisateur ne peut charger cet enregistrement pour modification. Les autres utilisateurs peuvent toujours, cependant, ajouter des enregistrements dans la table, soit manuellement en mode Utilisation, soit par les commandes CREER ENREGISTREMENT ou AJOUTER ENREGISTREMENT.

Le mode lecture/écriture est le mode par défaut pour toutes les tables quand une base est ouverte et un nouveau process démarré.

Changer l'état d'une table

Vous pouvez utiliser les commandes LECTURE SEULEMENT et LECTURE ECRITURE pour changer l'état d'une table. Si vous voulez changer l'état d'une table pour mettre un enregistrement en lecture seulement ou lecture/écriture, vous devez exécuter la commande avant le chargement de l'enregistrement. Un enregistrement déjà chargé n'est pas affecté par les commandes LECTURE SEULEMENT et LECTURE ECRITURE.

Chaque process dispose de son propre état (lecture seulement ou lecture/écriture) pour chaque table dans la base.

Charger, modifier et libérer des enregistrements

Pour que l'utilisateur local puisse modifier un enregistrement, la table doit être en mode lecture/écriture et l'enregistrement doit être chargé et non verrouillé.

Chacune des commandes chargeant un enregistrement courant (s'il y en a un) — telles que ENREGISTREMENT SUIVANT, CHERCHER, TRIER, CHARGER SUR LIEN, etc. — influe sur l'état de l'enregistrement. L'enregistrement est chargé en fonction de l'état courant de sa table (lecture seulement ou lecture/écriture) et sa propre disponibilité. Un enregistrement peut aussi être chargé d'une table liée par une commande activant le lien automatique.

Lorsqu'une table est en mode lecture seulement, tout enregistrement chargé de cette table est verrouillé. Un enregistrement verrouillé ne peut être sauvegardé ou supprimé. Le mode lecture seulement est le mode recommandé puisqu'il autorise les autres utilisateurs à charger, modifier, et ensuite sauvegarder l'enregistrement.

Lorsqu'une table est en mode lecture/écriture, tout enregistrement chargé de cette table est déverrouillé seulement si aucun autre utilisateur ne l'a préalablement verrouillé. Un enregistrement déverrouillé peut être modifié et sauvegardé. Une table doit être placée en mode lecture/écriture avant qu'on ait besoin de charger, modifier et sauvegarder l'enregistrement.

Si un enregistrement doit être modifié, utilisez la fonction Enregistrement verrouille pour tester s'il est ou non verrouillé par un autre utilisateur. Si l'enregistrement est verrouillé (Enregistrement verrouille retourne Vrai), chargez l'enregistrement avec la commande CHARGER ENREGISTREMENT et testez de nouveau le verrouillage. Répétez l'opération jusqu'à ce que l'enregistrement soit libéré (Enregistrement verrouille retourne Faux).

Lorsque vous en avez terminé avec les modifications à apporter à un enregistrement, il doit être libéré pour les autres utilisateurs (et donc déverrouillé) avec la commande LIBERER ENREGISTREMENT. Si un enregistrement n'est pas libéré, il reste verrouillé pour tous les autres utilisateurs jusqu'à ce qu'un nouvel enregistrement courant soit sélectionné. Changer l'enregistrement courant d'une table libère automatiquement l'enregistrement courant précédent. Vous devez explicitement appeler LIBERER ENREGISTREMENT si vous ne changez pas l'enregistrement courant. Ce principe ne s'applique qu'aux enregistrements existants : quand un enregistrement est créé, il peut être sauvegardé quel que soit l'état de la table auquel il appartient.

Utilisez la commande VERROUILLE PAR pour savoir quel utilisateur ou process a verrouillé un enregistrement.

Boucles pour charger des enregistrements non verrouillés

Cet exemple présente la boucle la plus simple pour charger un enregistrement non verrouillé :

```
LECTURE ECRITURE ([Clients])  ` Mettre la table en lecture/écriture
Repeter  ` Boucler jusqu'à ce que l'enregistrement soit déverrouillé
   CHARGER ENREGISTREMENT ([Clients])  ` Charger et verrouiller l'enregistrement
Jusque (Non (Enregistrement verrouille([Clients])))
```

La boucle s'exécute jusqu'à ce que l'enregistrement soit libéré.

Une boucle de ce type ne s'emploie que lorsqu'il est peu probable que l'enregistrement soit verrouillé par quelqu'un d'autre, puisque l'utilisateur aurait à attendre la fin de la boucle. Aussi, une telle boucle est rarement utilisée, sauf si l'enregistrement n'est modifiable que par méthode.

Cet exemple utilise la boucle précédente pour charger un enregistrement non verrouillé et modifier l'enregistrement :

```
LECTURE ECRITURE([Stocks])
Repeter   ` Boucle jusqu'à ce que l'enregistrement soit déverrouillé
   CHARGER ENREGISTREMENT([Stocks])   ` Charger l'enregistrement et le verrouiller
Jusque (Non (Enregistrement verrouille([Stocks])))
[Stocks]Part Qté := [Stocks]Part Qté – 1   ` Modifier l'enregistrement
STOCKER ENREGISTREMENT ([Stocks])   ` Sauvegarder l'enregistrement
LIBERER ENREGISTREMENT ([Stocks])   ` Laisser d'autres utilisateurs le modifier
LECTURE SEULEMENT([Stocks])
```

La commande MODIFIER ENREGISTREMENT prévient automatiquement l'utilisateur si un enregistrement est verrouillé, et interdit sa modification. L'exemple suivant évite la notification automatique par un test préalable de l'enregistrement avec la fonction Enregistrement verrouille. Si l'enregistrement est verrouillé, l'utilisateur peut annuler.

Cet exemple teste si l'enregistrement courant est verrouillé pour la table [Commandes]. Si c'est le cas, le process est endormi par la méthode pendant quelques instants. Cette technique peut être utlisée à la fois dans un développement multi-utilisateurs et multi-process :

```
Repeter
      ` Vous n'avez pas besoin de lecture/écriture pour le moment
   LECTURE SEULEMENT([Commandes])
   CHERCHER([Commandes])
      ` Si la recherche est terminée et que des enregistrements sont retournés
   Si ((OK=1) & (Enregistrements trouves([Commandes])>0))
      LECTURE ECRITURE([Commandes]) ` Mettre la table en mode lecture/écriture
      CHARGER ENREGISTREMENT([Commandes])
      Tant que (Enregistrement verrouille([Commandes]) & (OK=1))
            ` Si l'enregistrement est verrouillé,
            ` boucler jusqu'à ce que l'enregistrement soit libéré
            ` Par qui l'enregistrement est-il verrouillé ?
         VERROUILLE PAR([Commandes];$Process;$Utilisateur;$Machine;$Nom)
         Si ($Process=-1) ` L'enregistrement a-t-il été détruit?
            ALERTE("L'enregistrement a été détruit dans l'intervalle.")
            OK:=0
         Sinon
            Si ($Utilisateur="") ` Etes en mode simple utilisateur ?
               $Utilisateur:="vous-même"
            Fin de si
            CONFIRMER("L'enregistrement est utilisé par "+$Utilisateur+
                                                  " dans le "+$Nom+" Process.")
```

```
            Si (OK=1) ` Si vous voulez attendre quelques secondes
               ENDORMIR PROCESS(Numero du process courant;120)  `Attente
                  ` Essayez de charger l'enregistrement
               CHARGER ENREGISTREMENT([Commandes])
            Fin de si
         Fin de si
      Fin tant que
      Si (OK=1)   ` L'enregistrement est libéré
            ` Vous pouvez modifier l'enregistrement
         MODIFIER ENREGISTREMENT([Commandes])
         LIBERER ENREGISTREMENT([Commandes])
      Fin de si
      LECTURE SEULEMENT([Commandes]) ` Retour à lecture seulement
      OK:=1
   Fin de si
Jusque (OK=0)
```

Comportement des commandes en cas d'enregistrement verrouillé

Certaines commandes du langage effectuent des actions particulières lorsqu'elles rencontrent un enregistrement verrouillé. Voici la liste de ces commandes :

• MODIFIER ENREGISTREMENT : Cette commande affiche une boîte de dialogue indiquant que l'enregistrement est utilisé. L'enregistrement n'est pas affiché et donc l'utilisateur ne peut le modifier. En mode Utilisation, l'enregistrement est visible en lecture seulement.
• MODIFIER SELECTION : Cette commande se comporte normalement à ceci près que lorsque l'utilisateur double-clique sur un enregistrement pour le modifier, MODIFIER SELECTION affiche une boîte de dialogue indiquant que l'enregistrement est utilisé et n'autorise que sa lecture.
• APPLIQUER A SELECTION : Cette commande charge un enregistrement verrouillé, mais ne le modifie pas. APPLIQUER A SELECTION peut être utilisée pour lire les informations de la table sans problème. Si la commande rencontre un enregistrement verrouillé, celui-ci est placé dans l'ensemble système LockedSet.
• SUPPRIMER SELECTION : Cette commande ne supprime pas l'enregistrement verrouillé, elle l'ignore simplement. L'enregistrement verrouillé est placé dans l'ensemble LockedSet.
• SUPPRIMER ENREGISTREMENT : Cette commande est ignorée si l'enregistrement est verrouillé. Aucune erreur n'est retournée. Vous devez vous assurer que l'enregistrement est non verrouillé avant d'exécuter cette commande.
• STOCKER ENREGISTREMENT : Cette commande est ignorée si l'enregistrement est verrouillé. Aucune erreur n'est retournée. Vous devez vous assurer que l'enregistrement est non verrouillé avant d'exécuter cette commande.

• TABLEAU VERS SELECTION : Cette commande ne sauvegarde pas les enregistrements verrouillés. Si elle rencontre un enregistrement verrouillé, celui-ci est placé dans l'ensemble LockedSet.
• ALLER A ENREGISTREMENT : Dans une base multi-utilisateurs/multi-process, des enregistrements peuvent être ajoutés ou supprimés par d'autres utilisateurs/process. Les numéros d'enregistrement peuvent donc varier. Soyez prudent lorsque vous référencez un enregistrement par son numéro dans une base multi-utilisateurs.
• Ensembles : Soyez prudent lors de la manipulation d'ensembles puisque les informations sur lesquelles était basée la construction de l'ensemble peuvent avoir été modifiées par un autre utilisateur ou process. Pour plus d'informations sur les ensembles, reportez-vous à la section Présentation des ensembles.

Référence

CHARGER ENREGISTREMENT, Enregistrement verrouille, Etat lecture seulement, LECTURE ECRITURE, LECTURE SEULEMENT, LIBERER ENREGISTREMENT, Méthodes, Variables, VERROUILLE PAR.

LECTURE ECRITURE {(table | *)}

Paramètre	Type		Description
table | *	Table	→	Table à définir en mode lecture/écriture ou * pour toutes les tables ou Table par défaut si ce paramètre est omis

Description

LECTURE ECRITURE place table en mode lecture/écriture pour le process dans lequel la commande a été appelée. Si vous passez le paramètre optionnel *, toutes les tables sont placées en mode lecture/écriture.

Après un appel à LECTURE ECRITURE, lorsqu'un enregistrement est chargé, il n'est pas verrouillé — sauf si un autre utilisateur l'a déjà chargé. Cette commande ne modifie pas le statut des enregistrements déjà chargés, seuls les enregistrements chargés par la suite sont affectés.

Par défaut, toutes les tables sont en mode lecture/écriture.

Utilisez LECTURE ECRITURE lorsque vous devez modifier un enregistrement et sauvegarder les modifications. Vous pouvez également appeler cette commande lorsque vous voulez qu'un enregistrement soit verrouillé pour les autres utilisateurs, même si vous ne souhaitez pas effectuer de modifications. Placer une table en mode lecture/écriture vous permet d'empêcher les autres utilisateurs d'effectuer des modifications sur cette table. Cependant, ils peuvent continuer à créer des nouveaux enregistrements.

Note : Cette commande n'est pas rétroactive. Les privilèges de lecture/écriture pour un enregistrement sont définis par ceux de la table au moment où l'enregistrement est chargé. Pour qu'un enregistrement soit chargé en mode lecture/écriture alors que la table est en mode lecture seulement, vous devez placer la table en mode lecture/écriture avant que l'enregistrement soit chargé.

Référence

Etat lecture seulement, LECTURE SEULEMENT, Verrouillage d'enregistrements.

LECTURE SEULEMENT {(table | *)}

Paramètre	Type		Description
table | *	Table	→	Table à définir en mode lecture seulement ou * pour toutes les tables ou Table par défaut si ce paramètre est omis

Description

LECTURE SEULEMENT place table en mode lecture seulement pour le process dans lequel la commande a été appelée. Tous les enregistrements chargés par la suite sont verrouillés, aucune modification ne peut leur être apportée. Si vous passez le paramètre optionnel *, toutes les tables sont placées en mode lecture seulement.

Vous pouvez utiliser LECTURE SEULEMENT lorsqu'il n'est pas utile de modifier les enregistrements.

Note : Cette commande n'est pas rétroactive. Les privilèges de lecture/écriture pour un enregistrement sont définis par ceux de la table au moment où l'enregistrement est chargé. Pour qu'un enregistrement soit chargé en mode lecture seulement alors que la table est en mode lecture/écriture, vous devez placer la table en mode lecture seulement avant que l'enregistrement soit chargé.

Référence

Etat lecture seulement, LECTURE ECRITURE, Verrouillage d'enregistrements.

Etat lecture seulement {(table)} → Booléen

Paramètre	Type		Description
table	Table	→	Table pour laquelle il faut tester l'état ou Table par défaut si ce paramètre est omis
Résultat	Booléen	←	Accès à la table est lecture seulement (Vrai) ou Accès à la table est lecture-écriture (Faux)

Description

La fonction Etat lecture seulement est utilisée pour tester si table est en mode lecture seulement dans le process où la fonction est appelée. Etat lecture seulement retourne Vrai si table est en lecture seulement, et Faux si table est en lecture-écriture.

Exemple

L'exemple suivant teste le statut de la table [Factures]. Si elle est en lecture seulement, le mode lecture-écriture lui est appliqué et l'enregistrement courant est rechargé.

```
⇒   Si (Etat lecture seulement([Factures]))
       LECTURE ECRITURE([Factures])
       CHARGER ENREGISTREMENT([Factures])
    Fin de si
```

Note : L'enregistrement courant est rechargé pour permettre à l'utilisateur de le modifier. En effet, un enregistrement précédemment chargé en mode lecture seulement reste verrouillé jusqu'à ce qu'il soit rechargé en mode lecture-écriture.

Référence

LECTURE ECRITURE, LECTURE SEULEMENT, Verrouillage d'enregistrements.

CHARGER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle charger l'enregistrement courant ou Table par défaut si ce paramètre est omis

Description

CHARGER ENREGISTREMENT charge l'enregistrement courant de table. S'il n'y a pas d'enregistrement courant, CHARGER ENREGISTREMENT ne fait rien.

Il est alors utile d'appeler la fonction Enregistrement verrouille pour déterminer si l'enregistrement peut être modifié :
• Si la table est en mode Lecture seulement, la fonction retourne VRAI et vous ne pouvez pas modifier l'enregistrement.
• Si la table est en mode Lecture/écriture mais si l'enregistrement est déjà verrouillé, il est en mode Lecture seulement et vous ne pouvez pas le modifier.
• Si la table est en mode Lecture/écriture et si l'enregistrement n'est pas verrouillé, vous pouvez le modifier dans le process courant. La fonction Enregistrement verrouille retournera VRAI pour tous les autres utilisateurs et process.

Note : Si la commande LIRE VARIABLES est exécutée après un LECTURE SEULEMENT, l'enregistrement est automatiquement libéré et chargé, sans qu'il soit nécessaire d'appeler la commande LIBERER ENREGISTREMENT.

Vous n'aurez normalement pas besoin d'appeler la commande CHARGER ENREGISTREMENT, car toutes les commandes telles que CHERCHER, ENREGISTREMENT SUIVANT, ENREGISTREMENT PRECEDENT, etc., chargent automatiquement l'enregistrement courant.

En environnements multi-utilisateurs et multi-process, lorsque vous devez modifier un enregistrement existant, il vous faut accéder en Lecture/écriture à la table à laquelle appartient l'enregistrement. Lorsqu'un enregistrement verrouillé ne peut être chargé, CHARGER ENREGISTREMENT vous permet de tenter à nouveau plus tard de charger l'enregistrement. En utilisant CHARGER ENREGISTREMENT dans une boucle, vous pouvez attendre que l'enregistrement devienne accessible en Lecture/écriture.

Référence

Enregistrement verrouille, LIBERER ENREGISTREMENT, Verrouillage d'enregistrements.

LIBERER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table pour laquelle l'enregistrement est à libérer ou Table par défaut si ce paramètre est omis

Description

LIBERER ENREGISTREMENT place l'enregistrement courant de table dans l'état non verrouillé.

Si l'enregistrement n'est pas verrouillé par un utilisateur (mais est verrouillé pour les utilisateurs), LIBERER ENREGISTREMENT libère l'enregistrement pour tous les utilisateurs.

Même si LIBERER ENREGISTREMENT libère l'enregistrement de la mémoire, celui-ci reste l'enregistrement courant. Lorsqu'un enregistrement devient l'enregistrement courant, le précédent est automatiquement libéré et n'est donc plus verrouillé pour les autres utilisateurs. Vous devez appeler cette commande lorsque vous avez fini de modifier un enregistrement, que vous voulez qu'il reste l'enregistrement courant et qu'il soit accessible aux autres utilisateurs.

Si les enregistrements contiennent une quantité importante de données, de champs Image ou de documents externes (tels que des documents 4D Write ou 4D Draw), il est préférable de ne pas stocker l'enregistrement courant en mémoire sauf si vous avez besoin de le modifier. Dans ce cas, il faut utiliser la commande LIBERER ENREGISTREMENT. De cette manière, vous pouvez conserver l'enregistrement courant sans qu'il soit en mémoire. Vous libérez ainsi la mémoire occupée par l'enregistrement, mais vous n'avez pas accès aux valeurs stockées dans les champs. Si vous avez besoin d'exploiter ces valeurs, il faut utiliser la commande CHARGER ENREGISTREMENT.

Référence

CHARGER ENREGISTREMENT, Verrouillage d'enregistrements.

Enregistrement verrouille {(table)} → Booléen

Paramètre	Type		Description
table	Table	→	Table de l'enregistrement dont vous voulez tester le verrouillage ou Table par défaut si ce paramètre est omis
Résultat	Booléen	←	L'enregistrement est verrouillé (Vrai) ou L'enregistrement n'est pas verrouillé (Faux)

Description

Enregistrement verrouille teste si l'enregistrement courant de table est verrouillé. Cette fonction vous permet de savoir si un enregistrement est verrouillé ou non, et donc de réagir de manière appropriée, par exemple en laissant à l'utilisateur le choix d'attendre que l'enregistrement soit libéré ou d'annuler l'opération.

Si Enregistrement verrouille retourne Vrai, l'enregistrement est verrouillé par un autre utilisateur ou un autre process et ne peut être sauvegardé. Dans ce cas, vous devez appeler la commande CHARGER ENREGISTREMENT pour tenter à nouveau de charger l'enregistrement, jusqu'à ce que Enregistrement verrouille retourne Faux.

Si Enregistrement verrouille retourne Faux, l'enregistrement n'est pas verrouillé, ce qui signifie qu'il est verrouillé pour tous les autres utilisateurs. Seul l'utilisateur ayant chargé l'enregistrement ou le process courant peut modifier et sauvegarder l'enregistrement. Une table doit être en mode lecture/écriture si vous voulez modifier les enregistrements qu'elle contient.

Lorsque vous tentez de charger un enregistrement qui a été supprimé, Enregistrement verrouille continue de retourner Vrai. Pour éviter d'attendre un enregistrement qui n'existe plus, appelez la commande VERROUILLE PAR. Cette commande retourne -1 dans le paramètre Process si l'enregistrement a été supprimé.

Au cours d'une transaction, CHARGER ENREGISTREMENT et Enregistrement verrouille sont souvent appelées pour tester la disponibilité des enregistrements. Si un enregistrement est verrouillé, il suffit d'annuler la transaction.

Référence

CHARGER ENREGISTREMENT, Verrouillage d'enregistrements, VERROUILLE PAR.

VERROUILLE PAR ({table; }process; utilisateur; machine; nomProcess)

Paramètre	Type		Description
table	Table	→	Table de l'enregistrement verrouillé ou Table par défaut si ce paramètre est omis
process	Numérique	←	Numéro du process sur le serveur
utilisateur	Alpha	←	Nom de l'utilisateur
machine	Alpha	←	Nom de la machine
nomProcess	Alpha	←	Nom du process

Description

VERROUILLE PAR retourne des informations sur l'utilisateur et le process qui ont verrouillé l'enregistrement. Le numéro du process (sur le poste serveur), le nom de l'utilisateur, le possesseur et le nom du process sont retournés dans les variables process, utilisateur, machine et nomProcess. Vous pouvez utiliser ces informations dans une boîte de dialogue pour avertir l'utilisateur lorsqu'un enregistrement est verrouillé.

Si l'enregistrement n'est pas verrouillé, process prend la valeur 0 et utilisateur, machine et nomProcess retournent des chaînes vides. Si vous essayez de charger en lecture/écriture un enregistrement qui a été supprimé, process retourne -1 et utilisateur, machine et nomProcess retournent des chaînes vides.

En mode mono-utilisateur, cette commande retourne des valeurs dans process et nomProcess seulement si un enregistrement est verrouillé. Les valeurs retournées dans utilisateur et machine sont, dans ce cas, des chaînes vides.

Le paramètre utilisateur est le nom de l'utilisateur défini dans l'éditeur de mots de passe de 4e Dimension. Si aucun mot de passe n'a été défini, "Super_Utilisateur" est retourné.

Le paramètre machine retourné correspond au nom du possesseur de la machine tel qu'il est défini dans la page Identification du panneau de configuration Réseau (Windows) ou dans le tableau de bord Réglages partage de fichiers (MacOS). Toute modification du nom n'est prise en compte qu'après le redémarrage de la machine.

Référence

Enregistrement verrouille, Verrouillage d'enregistrements.

15

Ensembles

Présentation des ensembles

version 6.0 (Modifiée)

Les ensembles sont un moyen puissant et rapide de manipuler des sélections d'enregistrements. En plus de la possibilité de créer des ensembles, de les relier à la sélection courante, de les stocker, de les charger et de les effacer, 4D permet d'effectuer trois opérations standard sur les ensembles :
• Intersection,
• Union,
• Différence.

Ensembles et sélection courante

Un ensemble est une représentation d'une sélection d'enregistrements. L'idée d'ensemble est intimement liée à celle de sélection courante. Les ensembles sont généralement utilisés pour les raisons suivantes :
• Sauvegarder et ensuite restaurer une sélection lorsque l'ordre n'a pas d'importance,
• Accéder à la sélection que l'utilisateur a faite à l'écran (l'ensemble UserSet),
• Réaliser une opération logique entre des sélections.

La sélection courante est une liste de références qui pointent vers chaque enregistrement actuellement sélectionné. La liste existe en mémoire. Seuls les enregistrements sélectionnés sont dans la liste. Une sélection ne contient pas en réalité les enregistrements, mais seulement une liste de références à ces enregistrements. Chaque référence à un enregistrement utilise 4 octets en mémoire. Lorsque vous travaillez sur une table, vous travaillez toujours avec les enregistrements de la sélection courante. Lorsqu'une sélection est triée, seule la liste des références est réorganisée. Il n'y a qu'une sélection courante pour chaque table dans un process.

Comme une sélection courante, un ensemble représente une sélection d'enregistrements. Un ensemble le fait en utilisant une représentation très compacte pour chaque enregistrement. En effet, chacun est représenté par un bit (un huitième d'octet). Les opérations utilisant les ensembles sont très rapides parce les ordinateurs accomplissent très rapidement les opérations sur les bits. Un ensemble contient un bit pour chaque enregistrement de la table, que l'enregistrement soit inclus dans l'ensemble ou non. En fait, chaque bit est égal à 1 ou 0 — tout dépend si l'enregistrement est dans l'ensemble ou non.

Les ensembles sont très économiques en termes de mémoire RAM. La taille d'un ensemble, en octets, est toujours égale au nombre total des enregistrements dans la table divisé par huit. Ainsi, pour une table contenant 10 000 enregistrements, l'ensemble prend 1250 octets, ce qui fait environ 1,2 Ko de RAM.

Il peut exister plusieurs ensembles pour chaque table. En fait, les ensembles peuvent être sauvegardés sur disque indépendamment de la base. Pour modifier un enregistrement appartenant à un ensemble, vous devez d'abord utiliser l'ensemble comme sélection courante, puis modifier l'enregistrement. Le nom d'un ensemble interprocess doit être unique dans la base.

Un ensemble n'est jamais trié — les enregistrements sont simplement marqués comme appartenant ou non à l'ensemble. En revanche, une sélection temporaire est triée, mais requiert davantage de mémoire dans la plupart des cas. Pour plus d'information sur les sélections temporaires, reportez-vous à la section Sélections temporaires.

Au moment de sa création, un ensemble "mémorise" l'enregistrement courant de la sélection.

Comparaison	Sélection courante	Ensembles
Nombre par table	1	illimité
Tri	Oui	Non
Sauvegarde sur disque	Non	Oui
RAM par enreg. (en octets)	Nb enreg. sélec*4	Nb total enreg./8
Opérations	Non	Oui
Contient enreg. courant	Oui	Oui, celui du moment de création

L'ensemble que vous créez appartient à la table dans laquelle il a été créé. Les opérations sur les ensembles ne peuvent être effectuées qu'entre ensembles appartenant à la même table.

Les ensembles sont indépendants des données, ce qui signifie qu'après des modifications dans une table, un ensemble peut n'être plus exact. Bien des opérations peuvent rendre un ensemble inexact. Si vous créez un ensemble de tous les habitants de New York et changez ensuite les données de l'un des enregistrements par "New Jersey," l'ensemble ne change pas puisqu'il est simplement la représentation d'une sélection d'enregistrements. L'ajout ou la suppression d'enregistrements peut également rendre un ensemble obsolète. Les ensembles ne sont exacts que tant que la sélection d'origine n'est pas modifiée.

Ensembles process et interprocess

Vous pouvez utiliser trois types d'ensembles :

• **Ensembles process** : Un ensemble process n'est accessible que par le process dans lequel il a été créé. L'ensemble système LockedSet est un ensemble process. Les ensembles process sont effacés dès que la méthode du process est achevée. Les ensembles process ne requièrent pas de préfixe pour leur nom.

• **Ensembles interprocess** : Un ensemble est interprocess lorsque son nom est précédé des symboles (<>) — le signe “inférieur à” suivi du signe “supérieur à”. Cette syntaxe est utilisable à la fois sous Windows et MacOS. Sous MacOS, vous pouvez aussi utiliser le symbole “diamant” (Option+v sur un clavier français).

• **Ensembles locaux/Ensembles client** : La version 6 de 4D introduit la notion d'ensembles locaux/client. Leur nom est toujours précédé du symbole dollar ($) — à l'exception de l'ensemble système UserSet.

Ensembles et transactions

Un ensemble peut être créé à l'intérieur d'une transaction. Il est donc possible de définir un "ensemble des enregistrements créés pendant la transaction" et un "ensemble des enregistrements créés ou modifiés en-dehors de la transaction". Une fois la transaction terminée, l'ensemble créé pendant la transaction doit être effacé car il pourrait ne plus être une représentation exacte des enregistrements, surtout si la transaction a été annulée.

Exemple d'ensemble

Cet exemple détruit des enregistrements doublons dans une table. La table contient des informations sur des personnes. Une structure répétitive Boucle...Fin de boucle parcourt tous les enregistrements, comparant l'enregistrement courant à l'enregistrement précédent. Si le nom, l'adresse et le code postal sont identiques, l'enregistrement est ajouté à un ensemble. A la fin de la boucle, l'ensemble devient la sélection courante et l'ancienne sélection courante est détruite :

```
ENSEMBLE VIDE([Personnes];"Doublons")
   ` Créer un ensemble vide pour les doublons
TOUT SELECTIONNER([Personnes])
   ` Tous les enregistrements
   ` Trier les enregistrements par code postal, adresse et nom pour
   ` que les doublons soient les uns à côté des autres
TRIER ([Personnes];[Personnes]CODEPOSTAL;>;[Personnes]Adresse;>;[Personnes]Nom;>)
   ` Initialiser les variables conservant les champs des enregistrements précédents
$Nom:=[Personnes]Nom
$Adresse:=[Personnes]Adresse
$CODEPOSTAL:=[Personnes]CODEPOSTAL
   ` Aller au second enregistrement pour le comparer au premier
ENREGISTREMENT SUIVANT ([Personnes])
Boucle ($i; 2; Enregistrements dans table ([Personnes]))
      ` Parcourir les enregistrements en partant à 2
      ` Si nom, adresse et CODEPOSTAL sont identiques au
      ` précédent enregistrement, alors c'est un doublon.
```

```
   Si (([Personnes]Nom=$Nom) & ([Personnes]Adresse=$Adresse) &
                                        ([Personnes]CODEPOSTAL=$CODEPOSTAL))
      ` Ajouter enregistrement (le doublon) à l'ensemble
      ADJOINDRE ELEMENT ([Personnes]; "Doublon")
   Sinon
      ` Garder les nom, adresse et CODEPOSTAL de cet enregistrement pour
      ` comparer avec le prochain
      $Nom:=[Personnes]Nom
      $Adresse:=[Personnes]Adresse
      $CODEPOSTAL:=[Personnes]CODEPOSTAL
   Fin de si
      ` Passer à l'enregistrement suivant
   ENREGISTREMENT SUIVANT([Personnes])
Fin de boucle
   ` Utiliser doublons trouvés (en tant que sélection courante)
UTILISER ENSEMBLE ("Doublons")
   ` Détruire doublons
SUPPRIMER SELECTION ([Personnes])
   ` Supprimer l'ensemble de la mémoire
EFFACER ENSEMBLE ("Doublons")
```

Au lieu de supprimer immédiatement les enregistrements à la fin de la méthode, vous pouvez les afficher à l'écran ou les imprimer si des comparaisons plus fines doivent être menées.

Ensemble système UserSet

4D gère un ensemble système local/client nommé UserSet. UserSet stocke automatiquement la sélection d'enregistrements la plus récemment effectuée à l'écran par l'utilisateur. Ainsi, vous pouvez afficher un groupe d'enregistrements avec MODIFIER SELECTION ou VISUALISER SELECTION, demander à l'utilisateur d'en sélectionner certains et retourner le résultat de cette sélection manuelle dans un ensemble que vous nommez ou dans une sélection.

4D Server : Bien que son nom ne débute pas par le caractère "$", l'ensemble système UserSet est un ensemble client. Par conséquent, lors de l'utilisation des commandes INTERSECTION, REUNION et DIFFERENCE, veillez à ne comparer UserSet qu'à d'autres ensembles clients. Pour plus d'informations, reportez-vous aux descriptions de ces commandes ainsi qu'à la section 4D Server et les ensembles dans le *Guide de référence de 4D Server*.

Les tables ne disposent pas de leurs propres UserSet. Un UserSet n'"appartient" à une table que lorsqu'une sélection d'enregistrements est affichée pour cette table.

La méthode ci-dessous illustre comment afficher des enregistrements pour permettre à l'utilisateur d'en sélectionner quelques-uns, et ensuite utiliser le UserSet pour afficher les enregistrements sélectionnés :

```
` Afficher tous les enregistrements et permettre à l'utilisateur d'en sélectionner un
` certain nombre.
` Puis afficher cette sélection en utilisant UserSet pour modifier la sélection courante.
FORMULAIRE SORTIE([Personnes]; "Display") ` Choisir le formulaire sortie
TOUT SELECTIONNER ([Personnes]) ` Sélection de toutes les personnes
ALERTE ("Appuyer Ctrl ou Commande + Clic pour sélectionner des enregistrements.")
VISUALISER SELECTION ([Personnes]) ` Afficher les personnes
UTILISER ENSEMBLE ("UserSet") ` Utiliser les personnes sélectionnées
ALERTE ("Vous avez choisi les personnes suivantes.")
VISUALISER SELECTION ([Personnes]) ` Afficher les personnes sélectionnées
```

Note : Vous devez exécuter MODIFIER SELECTION ou VISUALISER SELECTION pour récupérer le UserSet.

Ensemble système LockedSet

Les commandes APPLIQUER A SELECTION, TABLEAU VERS SELECTION et SUPPRIMER SELECTION créent un ensemble système nommé LockedSet lorsqu'elles sont utilisées en environnement multiprocess. LockedSet indique quels enregistrements étaient verrouillés lors de l'exécution d'une commande.

Référence

ADJOINDRE ELEMENT, Appartient a ensemble, CHARGER ENSEMBLE, COPIER ENSEMBLE, DIFFERENCE, EFFACER ENSEMBLE, ENLEVER ELEMENT, Enregistrements dans ensemble, ENSEMBLE VIDE, INTERSECTION, NOMMER ENSEMBLE, REUNION, STOCKER ENSEMBLE, UTILISER ENSEMBLE.

ENSEMBLE VIDE ({table; }ensemble)

Paramètre	Type		Description
table	Table	→	Table pour laquelle créer un ensemble vide ou Table par défaut si ce paramètre est omis
ensemble	Alpha	→	Nom du nouvel ensemble vide

Description

ENSEMBLE VIDE crée un ensemble vide, ensemble, pour table. Vous pouvez ajouter des enregistrements dans cet ensemble à l'aide de la commande AJOINDRE ELEMENT. Si un ensemble du même nom existe déjà, il est effacé et remplacé par le nouvel ensemble.

Note : Il n'est pas indispensable d'appeler la commande ENSEMBLE VIDE avant d'utiliser la commande NOMMER ENSEMBLE.

Exemple

Reportez-vous à l'exemple présenté dans la section Présentation des ensembles.

Référence

EFFACER ENSEMBLE, NOMMER ENSEMBLE.

NOMMER ENSEMBLE ({table; }ensemble)

Paramètre	Type		Description
table	Table	→	Table pour laquelle vous voulez créer un ensemble à partir de la sélection courante ou Table par défaut si ce paramètre est omis
ensemble	Alpha	→	Nom du nouvel ensemble

Description

NOMMER ENSEMBLE crée un nouvel ensemble, ensemble, pour table, et y place la sélection courante. Le pointeur d'enregistrement courant de la table est sauvegardé avec ensemble. Si ensemble est passé à la commande UTILISER ENSEMBLE, la sélection courante et l'enregistrement courant sont restitués. Comme pour tout ensemble, il ne peut y avoir de tri, et lorsque ensemble est appelé, l'ordre par défaut est utilisé. Si un ensemble du même nom existe déjà, il est effacé et remplacé par le nouvel ensemble.

Exemple

L'exemple suivant crée un ensemble après qu'une recherche ait été effectuée, de manière à ce que l'ensemble puisse être stocké sur disque :

```
     CHERCHER ([Personnes])  ` L'utilisateur effectue une recherche
        ` Création d'un nouvel ensemble
⇒    NOMMER ENSEMBLE ([Personnes]; "EnsembleRecherche")
        ` L'ensemble est stocké sur disque
     STOCKER ENSEMBLE ("EnsembleRecherche"; "MaRecherche")
```

Référence

EFFACER ENSEMBLE, ENSEMBLE VIDE.

CREER ENSEMBLE SUR TABLEAU

Ensembles

version 6.7 (Modifiée)

CREER ENSEMBLE SUR TABLEAU (table; tabEnrg{; nomEns})

Paramètre	Type		Description
table	Table	→	Table de l'ensemble
tabEnrg	Tab Entier long | Tab Booléen	→	Tableau de n° d'enregistrements, ou Tableau de booléens (Vrai = l'enregistrement est dans l'ensemble, Faux = il n'est pas dans l'ensemble)
nomEns	Alpha	→	Nom de l'ensemble à créer, ou Appliquer la commande à l'ensemble Userset si ce paramètre est omis ou vide

Description

La commande CREER ENSEMBLE SUR TABLEAU crée l'ensemble nomEns à partir :
• soit du tableau de numéros d'enregistrements absolus tabEnrg de la table table,
• soit du tableau de booléens tabEnrg ; dans ce cas, les valeurs du tableau indiquent l'appartenance (VRAI) ou non (FAUX) de chaque enregistrement de table à l'ensemble nomEns.

Lorsque vous utilisez la commande avec un tableau d'entiers longs, tous les numéros du tableau représentent la liste des numéros d'enregistrements qui feront partie de l'ensemble nomEns. Si un numéro est invalide (enregistrement non créé), l'erreur -10503 est générée.

Lorsque vous utilisez la commande avec un tableau de booléens, le Nième élément du tableau représente l'intégration (VRAI) ou non (FAUX) de l'enregistrement numéro N à l'ensemble nomEns. En principe, le nombre d'éléments du tableau doit être égal au nombre d'enregistrements de table. Si le tableau est plus petit que le nombre d'enregistrements, seuls les enregistrements définis par le tableau pourront faire partie de l'ensemble.

Note : Avec un tableau de booléens, la commande utilise les éléments à partir du numéro 0 jusqu'au numéro N-1.

Si vous ne passez pas le paramètre nomEns ou si vous passez une chaîne vide, la commande s'applique à l'ensemble système Userset.

Référence

CREER SELECTION SUR TABLEAU, TABLEAU BOOLEEN SUR ENSEMBLE.

Gestion des erreurs

Dans un tableau d'entier longs, si un numéro d'enregistrement est invalide (enregistrement non créé), l'erreur -10503 est générée.

UTILISER ENSEMBLE (ensemble)

Paramètre	Type		Description
ensemble	Alpha	→	Nom de l'ensemble à utiliser

Description

UTILISER ENSEMBLE crée, avec les enregistrements de ensemble, une nouvelle sélection courante pour la table à laquelle ensemble appartient.

Au moment où vous créez un ensemble, la position de l'enregistrement courant est sauvegardée. UTILISER ENSEMBLE récupère cette information et fait de l'enregistrement le nouvel enregistrement courant. Si vous supprimez cet enregistrement avant d'exécuter UTILISER ENSEMBLE, 4e Dimension sélectionne comme enregistrement courant le premier enregistrement de l'ensemble. Les commandes du thème "Ensembles", INTERSECTION, REUNION, DIFFERENCE, et ADJOINDRE ELEMENT réinitialisent l'enregistrement courant. Si vous avez créé un ensemble ne contenant pas de position d'enregistrement courant, UTILISER ENSEMBLE désigne le premier enregistrement de l'ensemble comme enregistrement courant.

ATTENTION : Rappelez-vous qu'un ensemble est la représentation d'une sélection d'enregistrements à un instant donné (au moment de sa création). Si les enregistrements que l'ensemble représente sont modifiés, il se peut que celui-ci ne soit plus valide. En conséquence, un ensemble sauvegardé sur disque doit généralement représenter un groupe d'enregistrements qui ne change pas souvent. De multiples événements peuvent rendre un ensemble invalide, par exemple la suppression ou la modification d'un enregistrement, ou encore la modification des critères de création de l'ensemble.

Exemples

L'exemple suivant utilise CHARGER ENSEMBLE pour charger un ensemble des sites de la société Dubois à Paris. UTILISER ENSEMBLE est ensuite appelée pour faire de l'ensemble la sélection courante :

```
   ` Charger l'ensemble en mémoire
   CHARGER ENSEMBLE ([Entreprises]; "DuboisParis"; "ENSDuboisParis")
⇒  UTILISER ENSEMBLE ("DuboisParis") ` Modification de la sélection courante
   EFFACER ENSEMBLE ("DuboisParis") ` Effacement de l'ensemble de la mémoire
```

Référence

CHARGER ENSEMBLE, EFFACER ENSEMBLE.

ADJOINDRE ELEMENT ({table; }ensemble)

Paramètre	Type		Description
table	Table	→	Table de l'enregistrement courant ou Table par défaut si ce paramètre est omis
ensemble	Alpha	→	Nom de l'ensemble auquel ajouter l'enregistrement courant

Description

ADJOINDRE ELEMENT ajoute l'enregistrement courant de table à ensemble. L'ensemble doit avoir déjà été créé ; si ensemble n'existe pas, une erreur est retournée. S'il n'y a pas d'enregistrement courant pour table, ADJOINDRE ELEMENT ne fait rien.

Référence

ENLEVER ELEMENT.

ENLEVER ELEMENT ({table; }ensemble)

Paramètre	Type		Description
table	Table	→	Table de l'enregistrement courant ou Table par défaut si ce paramètre est omis
ensemble	Alpha	→	Nom de l'ensemble duquel supprimer l'enregistrement courant

Description

ENLEVER ELEMENT supprime l'enregistrement courant de table de l'ensemble ensemble. L'ensemble doit déjà exister ; s'il n'existe pas, une erreur est générée. S'il n'y a pas d'enregistrement courant dans table, ENLEVER ELEMENT ne fait rien.

Référence

ADJOINDRE ELEMENT.

EFFACER ENSEMBLE (ensemble)

Paramètre	Type		Description
ensemble	Alpha	→	Nom de l'ensemble à effacer de la mémoire

Description

EFFACER ENSEMBLE efface ensemble de la mémoire et la libère ainsi pour d'autres utilisations. EFFACER ENSEMBLE n'a aucune conséquence sur les tables, sélections ou enregistrements. Pour sauvegarder un ensemble avant de l'effacer, utiliser la commande STOCKER ENSEMBLE. Comme les ensembles consomment de la mémoire, pensez à les effacer dès qu'ils ne sont plus nécessaires.

Exemples

Reportez-vous à l'exemple de la commande UTILISER ENSEMBLE.

Référence

CHARGER ENSEMBLE, ENSEMBLE VIDE, NOMMER ENSEMBLE.

Appartient a ensemble

Ensembles

version 5

Appartient a ensemble (ensemble) → Booléen

Paramètre	Type		Description
ensemble	Alpha	→	Nom de l'ensemble à tester
Résultat	Booléen	←	L'enregistrement courant est dans l'ensemble (Vrai) ou l'enregistrement courant n'est pas dans l'ensemble (Faux)

Description

Appartient a ensemble teste si l'enregistrement courant de la table est inclus dans ensemble. La fonction Appartient a ensemble retourne Vrai si l'enregistrement courant de la table est dans ensemble, et retourne Faux si l'enregistrement courant de la table n'est pas dans ensemble.

Exemples

L'exemple suivant est la méthode objet d'un bouton testant si l'enregistrement courant est inclus dans l'ensemble des meilleurs clients :

```
⇒   Si (Appartient a ensemble ("Meilleurs"))
       ALERTE ("C'est un de nos meilleurs clients.")
    Sinon
       ALERTE ("Ce n'est pas un de nos meilleurs clients.")
    Fin de si
```

Référence

ADJOINDRE ELEMENT, ENLEVER ELEMENT.

Enregistrements dans ensemble (ensemble) → Numérique

Paramètre	Type		Description
ensemble	Alpha	→	Nom de l'ensemble à tester
Résultat	Numérique	←	Nombre d'enregistrements dans l'ensemble

Description

Enregistrements dans ensemble retourne le nombre d'enregistrements présents dans ensemble. Si ensemble n'existe pas ou s'il n'y a pas d'enregistrements dans ensemble, Enregistrements dans ensemble retourne 0.

Exemple

L'exemple suivant affiche une boîte de dialogue d'alerte qui indique le pourcentage des clients qui sont considérés comme les meillleurs :

```
      ` Calculer d'abord le pourcentage
⇒     $Pourcent := (Enregistrements dans ensemble ("Meilleurs") / Enregistrements
                                                    dans table([Clients])) * 100
      ` Afficher une alerte avec le pourcentage
      ALERTE (Chaine ($Pourcent; "##0%") + " de nos clients sont les meilleurs.")
```

Référence

Enregistrements dans table, Enregistrements trouves.

STOCKER ENSEMBLE (ensemble; document)

Paramètre	Type		Description
ensemble	Alpha	→	Nom de l'ensemble à stocker
document	Alpha	→	Nom du fichier dans lequel stocker l'ensemble

Description

STOCKER ENSEMBLE sauvegarde ensemble dans le fichier disque document.

Il n'est pas nécessaire que document ait le même nom que l'ensemble. Si vous passez une chaîne vide dans document, une boîte de dialogue standard de sauvegarde de fichiers apparaît, permettant à l'utilisateur de saisir un nom de fichier. Vous pourrez utiliser la commande CHARGER ENSEMBLE pour charger un ensemble stocké sur disque.

Si l'utilisateur clique sur le bouton Annuler dans la boîte de dialogue de sauvegarde de fichiers, ou si une erreur se produit lors de la sauvegarde, la variable système OK prend la valeur 0. Sinon, elle prend la valeur 1.

La commande STOCKER ENSEMBLE est souvent utilisée pour stocker sur disque les résultats d'une recherche particulièrement longue.

ATTENTION : Rappelez-vous qu'un ensemble est l'image d'une sélection d'enregistrements au moment précis où l'ensemble est créé. Si les enregistrements représentés par l'ensemble sont modifiés, celui-ci devient obsolète. En conséquence, vous devez créer et sauvegarder des ensembles représentant des enregistrements dont le contenu varie peu. De multiples événements peuvent rendre un ensemble obsolète : modification ou suppression d'un enregistrement de l'ensemble, ou encore modification des critères ayant déterminé la création de l'ensemble. Rappelez-vous également que les ensembles ne stockent pas les valeurs des champs.

Exemple

L'exemple suivant affiche la boîte de dialogue standard d'enregistrement de fichiers de manière à permettre à l'utilisateur de saisir le nom du fichier contenant l'ensemble :

```
⇒ STOCKER ENSEMBLE ("UnEnsemble"; "")
```

Référence

CHARGER ENSEMBLE.

Variables et ensembles système

Si l'utilisateur clique sur le bouton Annuler dans la boîte de dialogue standard de sauvegarde de documents, ou si une erreur se produit pendant la sauvegarde, la variable système OK prend la valeur 0. Sinon, elle prend la valeur 1.

CHARGER ENSEMBLE ({table; }ensemble; document)

Paramètre	Type		Description
table	Table	→	Table à laquelle appartient l'ensemble ou Table par défaut si ce paramètre est omis
ensemble	Alpha	→	Nom de l'ensemble à créer en mémoire
document	Alpha	→	Document disque contenant l'ensemble

Description

CHARGER ENSEMBLE charge un ensemble depuis le fichier document, créé à l'aide de la commande STOCKER ENSEMBLE.

L'ensemble stocké dans document doit s'appliquer à table. Si ensemble existait déjà en mémoire, il est réécrit.

Le paramètre document est le nom du fichier disque contenant l'ensemble. Il n'est pas nécessaire que ce fichier ait le même nom que l'ensemble. Si vous passez une chaîne vide dans document, une boîte de dialogue standard d'ouverture de fichiers s'affiche, permettant à l'utilisateur de choisir l'ensemble à charger.

ATTENTION : Rappelez-vous qu'un ensemble est l'image d'une sélection d'enregistrements au moment précis où l'ensemble est créé. Si les enregistrements représentés par l'ensemble sont modifiés, celui-ci devient obsolète. En conséquence, vous devez stocker et charger des ensembles avec des enregistrements dont le contenu varie peu. De multiples événements peuvent rendre un ensemble obsolète : modification ou suppression d'un enregistrement de l'ensemble, ou encore modification des critères ayant déterminé la création de l'ensemble.

Exemple

L'exemple suivant utilise CHARGER ENSEMBLE pour charger l'ensemble des locaux de l'entreprise Dupont SARL à Paris :

```
      ` Charger l'ensemble en mémoire
⇒     CHARGER ENSEMBLE ([Entreprises]; "Paris Dupont SARL"; "PaDupontEns")
         ` Modifier la sélection courante avec l'ensemble
      UTILISER ENSEMBLE ("Paris Dupont SARL")
      EFFACER ENSEMBLE ("Paris Dupont SARL") ` Effacer l'ensemble de la mémoire
```

Référence

STOCKER ENSEMBLE.

Variables et ensembles système

Si l'utilisateur clique sur Annuler dans la boîte de dialogue d'ouverture de fichiers, ou si une erreur se produit pendant le chargement, la variable système OK prend la valeur 0. Sinon, elle prend la valeur 1.

DIFFERENCE (ensemble1; ensemble2; résultat)

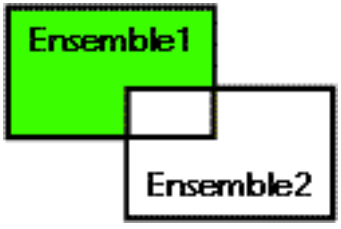
Paramètre	Type		Description
ensemble1	Alpha	→	Ensemble initial
ensemble2	Alpha	→	Ensemble à exclure
résultat	Alpha	→	Ensemble résultant

Description

DIFFERENCE fusionne ensemble1 et ensemble2 et exclut de l'ensemble résultat tous les enregistrements se trouvant dans ensemble2. Autrement dit, un enregistrement est inclus dans l'ensemble résultat s'il appartient à ensemble1 mais n'appartient pas à ensemble2. Le tableau suivant liste les résultats possibles d'une opération de différence d'ensembles.

Ensemble1	Ensemble2	Ensemble résultant
Oui	Non	Oui
Oui	Oui	Non
Non	Oui	Non
Non	Non	Non

Le schéma ci-dessous est la représentation graphique d'une opération de différence entre deux ensembles. La zone colorée est l'ensemble résultant.



L'ensemble résultat est créé par DIFFERENCE. Il remplace tout ensemble du même nom existant déjà, y compris ensemble1 et ensemble2. Les ensembles ensemble1 et ensemble2 doivent appartenir à la même table. L'ensemble résultat appartient à la même table que ensemble1 et ensemble2.

4D Server : En mode client/serveur, les ensembles interprocess et process sont conservés sur le poste serveur, alors que les ensembles locaux sont conservés sur les postes clients. DIFFERENCE requiert que les trois ensembles soient situés sur la même machine. Par conséquent, ils doivent tous être des ensembles locaux, ou bien aucun d'eux ne doit être local. Pour plus d'informations sur ce point, reportez-vous au paragraphe 4D Server et les ensembles dans le manuel de référence de 4D Server.

Exemples

L'exemple suivant exclut les enregistrements sélectionnés par l'utilisateur. Les enregistrements sont affichés à l'écran par l'instruction suivante :

```
VISUALISER SELECTION ([Clients]) ` Affichage des clients sous forme de liste
```

Un bouton associé à une méthode objet est placé en bas de la liste. La méthode objet exclut les enregistrements sélectionnés par l'utilisateur (l'ensemble système nommé UserSet) et affiche une sélection réduite :

```
    ` Création d'un ensemble à partir de la sélection courante
NOMMER ENSEMBLE ([Clients]; "$Courant")
    ` Exclusion des enregistrements sélectionnés
⇒ DIFFERENCE ("$Courant";"UserSet";"$Courant")
UTILISER ENSEMBLE ("$Courant") ` Utilisation du nouvel ensemble
EFFACER ENSEMBLE ("$Courant") ` Effacement de l'ensemble
```

Référence

INTERSECTION, REUNION.

INTERSECTION (ensemble1; ensemble2; résultat)

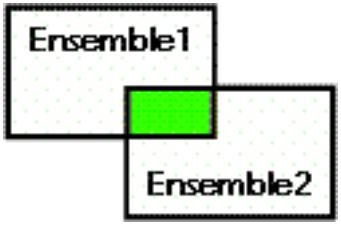
Paramètre	Type		Description
ensemble1	Alpha	→	Premier ensemble
ensemble2	Alpha	→	Second ensemble
résultat	Alpha	→	Ensemble résultant

Description

INTERSECTION compare ensemble1 et ensemble2 et sélectionne uniquement les enregistrements se trouvant à la fois dans ensemble1 et dans ensemble2. Le tableau suivant liste les résultats possibles d'une opération d'intersection d'ensembles.

Ensemble1	Ensemble2	Ensemble résultant
Oui	Non	Non
Oui	Oui	Oui
Non	Oui	Non
Non	Non	Non

Le schéma ci-dessous est la représentation graphique de l'intersection de deux ensembles. La zone colorée est l'ensemble résultant.



L'ensemble résultat est créé par INTERSECTION. Il remplace tout ensemble du même nom existant déjà, y compris ensemble1 et ensemble2. Les ensembles de départ ensemble1 et ensemble2 doivent appartenir à la même table. L'ensemble résultat appartient à la même table que ensemble1 et ensemble2.

4D Server : En mode client/serveur, les ensembles interprocess et process sont conservés sur le poste serveur, alors que les ensembles locaux sont conservés sur les postes clients. INTERSECTION requiert que les trois ensembles soient situés sur la même machine. Par conséquent, ils doivent tous être des ensembles locaux, ou bien aucun d'eux ne doit être local. Pour plus d'informations sur ce point, reportez-vous au paragraphe 4D Server et les ensembles dans le manuel de référence de 4D Server.

Exemple

L'exemple suivant recherche les clients en contact avec deux représentants, Jean et Grégoire. Chaque représentant dispose d'un ensemble regroupant ses clients. Les clients se trouvant dans les deux ensembles sont en contact avec Jean et Grégoire :

```
    ` Doublon reçoit les clients appartenant aux 2 ensembles
⇒   INTERSECTION ("Jean"; "Grégoire"; "Doublon")
    UTILISER ENSEMBLE ("Doublon") ` Modification de la sélection courante
    ` Effacement de cet ensemble mais sauvegarde des autres
    EFFACER ENSEMBLE ("Doublon")
    ` Affichage des clients en contact avec les deux commerciaux
    VISUALISER SELECTION ([Clients])
```

Référence

DIFFERENCE, REUNION.

REUNION (ensemble1; ensemble2; résultat)

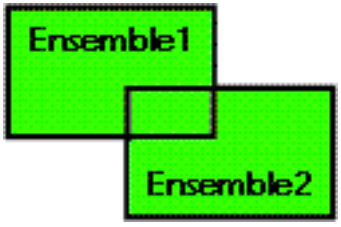
Paramètre	Type		Description
ensemble1	Alpha	→	Premier ensemble
ensemble2	Alpha	→	Second ensemble
résultat	Alpha	→	Ensemble résultant

Description

REUNION crée un nouvel ensemble contenant tous les enregistrements de ensemble1 et ensemble2. Le tableau suivant liste les résultats possibles d'une opération de réunion d'ensembles.

Ensemble1	Ensemble2	Ensemble résultant
Oui	Non	Oui
Oui	Oui	Oui
Non	Oui	Oui
Non	Non	Non

Le schéma ci-dessous est la représentation graphique de la réunion de deux ensembles. La zone colorée est l'ensemble résultant.



L'ensemble résultat est créé par REUNION. Il remplace tout ensemble du même nom existant déjà, y compris ensemble1 et ensemble2. Les ensembles de départ ensemble1 et ensemble2 doivent appartenir à la même table. L'ensemble résultat appartient à la même table que ensemble1 et ensemble2. L'enregistrement courant de résultat est celui de ensemble1.

4D Server : En mode client/serveur, les ensembles interprocess et process sont conservés sur le poste serveur, alors que les ensembles locaux sont conservés sur les postes clients. REUNION requiert que les trois ensembles soient situés sur la même machine. Par conséquent, ils doivent tous être des ensembles locaux, ou bien aucun d'eux ne doit être local. Pour plus d'informations sur ce point, reportez-vous au paragraphe 4D Server et les ensembles dans le manuel de référence de 4D Server.

Exemple

L'exemple suivant ajoute des enregistrements à l'ensemble des meilleurs clients. Les enregistrements sont affichés à l'écran. Ensuite, l'ensemble des meilleurs clients est chargé du disque, et tous les enregistrements sélectionnés par l'utilisateur (l'ensemble système UserSet) sont ajoutés. Enfin, le nouvel ensemble est sauvegardé sur le disque :

```
   TOUT SELECTIONNER ([Clients])  ` Sélection de tous les enregistrements
   VISUALISER SELECTION ([Clients])  ` Afficher tous les clients en mode liste
      ` Chargement de l'ensemble des meilleurs clients
   CHARGER ENSEMBLE ("$Meilleurs"; "$Meilleurs.sav")
⇒  REUNION ("$Meilleurs"; "UserSet"; "$Meilleurs")  ` Ajout de toute sélection à l'ensemble
      ` Sauvegarde de l'ensemble des meilleurs clients
   STOCKER ENSEMBLE ("$Meilleurs"; "$Meilleurs.sav")
```

Référence

DIFFERENCE, INTERSECTION.

COPIER ENSEMBLE (srcEns; dstEns)

Paramètre	Type		Description
srcEns	Alpha	→	Nom de l'ensemble source
dstEns	Alpha	→	Nom de l'ensemble de destination

Description

La commande COPIER ENSEMBLE copie le contenu de l'ensemble srcEns dans l'ensemble dstEns.

Les deux ensembles peuvent être process, interprocess ou locaux.

4D Server : En mode client/serveur, les ensembles interprocess et process sont conservés sur le poste serveur, alors que les ensembles locaux sont conservés sur les postes clients. COPIER ENSEMBLE vous permet de copier des ensembles d'un poste à l'autre. Pour plus d'informations sur ce point, reportez-vous au paragraphe 4D Server et les ensembles dans le manuel de référence de 4D Server.

Exemples

(1) L'exemple suivant, en client/serveur, copie l'ensemble local "$SetA", conservé sur le poste client, vers l'ensemble process "SetB", conservé sur le poste serveur :

⇒ **COPIER ENSEMBLE**("$SetA";"SetB")

(2) L'exemple suivant, en client/serveur, copie l'ensemble process "SetA", conservé sur le poste serveur, vers l'ensemble local "$SetB", conservé sur le poste client :

⇒ **COPIER ENSEMBLE**("SetA";"$SetB")

Référence

Présentation des ensembles.

16

Environnement 4D

Type application

Environnement 4D

version 6.0

Type application → Entier long

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	Valeur numérique représentant le type de l'application

Description

La fonction Type application renvoie une valeur numérique qui représente le type de l'environnement 4D que vous utilisez. 4D vous fournit les constantes prédéfinies suivantes :

Constante	Type	Valeur
4e Dimension	Entier long	0
4D Engine	Entier long	1
4D Runtime	Entier long	2
4D Runtime Classic	Entier long	3
4D Client	Entier long	4
4D Server	Entier long	5
4D First	Entier long	6

Exemple

Quelque part dans votre code, ailleurs que dans la méthode base Sur démarrage serveur, vous voulez vérifier si l'utilisateur a ouvert la base avec 4D Server. Pour cela, vous pouvez écrire les lignes de code suivantes :

```
⇒   Si (Type application=4D Server)
        Exécuter des actions nécessaires
    Fin de si
```

Référence

Type version, Version application.

Type version

Environnement 4D

version 6.0

Type version → Entier long

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	0 = Version standard 1 = Version de démonstration

Description

La commande Type version retourne une valeur numérique qui représente le type de version de 4D que vous utilisez. 4D vous fournit les constantes prédéfinies suivantes :

Constante	Type	Valeur
Version standard	Entier long	0
Version de démonstration	Entier long	1

Exemple

Votre application 4D inclut des fonctionnalités qui ne sont pas disponibles lorsque vous êtes en version démo. Vous pouvez tester l'environnement en écrivant le code suivant :

```
⇒   Si (Type version=Version standard)
       Exécuter les actions nécessaires
    Sinon
       ALERTE("Cette fonctionnalité n'est pas disponible dans la version démo de"
                                                          +"Super Gestion™.")
    Fin de si
```

Référence

Type application, Version application.

Version application

Environnement 4D

version 6.0

Version application {(*)} → Alpha

Paramètre	Type		Description
*	*	→	Si passé = numéro de version long Si omis = numéro de version court
Résultat	Alpha	←	Numéro de version dans une chaîne encodée

Description

Version application retourne une chaîne encodée qui exprime le numéro de version de l'environnement 4D que vous utilisez.

• Si vous ne passez pas le paramètre optionnel *, une chaîne de 4 caractères est retournée, formatée de la manière suivante :

Caractères	Description
1-2	Numéro de version
3	Numéro de mise à jour
4	Numéro de révision

Exemple : la chaîne "0600" représente la version 6.0.0.

• Si vous passez le paramètre optionnel *, une chaîne de 8 caractères est retournée, formatée de la manière suivante :

Caractères	Description
1	"F" représente une version finale "B" représente une version beta Les autres caractères représentent une version interne à 4D
2-3-4	Numéro de compilation interne à 4D
5-6	Numéro de version
7	Numéro de mise à jour
8	Numéro de révision

Exemple : la chaîne "B0120602" représente une version beta 12 de la version 6.0.2.

Exemples

(1) Cet exemple affiche le numéro de version de l'environnement 4D :

```
⇒    $vs4Dversion:=Version application
     ALERTE("Vous utilisez la version "+Chaine(Num(Sous chaine($vs4Dversion;1;2)))+"."+
$vs4Dversion≤3≥+"."+$vs4Dversion≤4≥)
```

(2) Cet exemple teste si vous utilisez une version finale :

```
⇒    Si(Sous chaine(Version application(*);1;1)#"F")
        ALERTE("Veuillez vous assurer que vous utilisez une version finale de 4D avec
                                                                    cette base !")
        QUITTER 4D
     Fin de si
```

Référence

Type application, Type version.

Application compilee

Environnement 4D

version 6.0

Application compilee → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Mode compilé (Vrai), mode interprété (Faux)

Description

La fonction Application compilee teste si la base tourne en mode compilé (Vrai) ou en mode interprété (Faux).

Exemple

Dans une de vos méthodes, vous avez placé du code pour déboguer la base lorsque vous êtes en mode interprété. Vous pouvez précéder ce code d'un test qui appelle la fonction Application compilee :

```
` ...
⇒ Si (Non(Application compilee))
     ` Mettre du code pour déboguer votre base ici
  Fin de si
  ` ...
```

Référence

APPELER 4D, Indefinie.

PROPRIETES PLATE FORME (plate-forme{; système{; machine}})

Paramètre	Type		Description
plate-forme	Numérique	←	1 = Macintosh 68K 2 = Power Macintosh 3 = Windows
système	Numérique	←	Dépend de la version que vous utilisez
machine	Numérique	←	Dépend de la version que vous utilisez

Description

La commande PROPRIETES PLATE FORME retourne des informations sur le type de plate-forme que vous utilisez, la version du système d'exploitation et le processeur installés.

PROPRIETES PLATE FORME retourne ces informations dans les paramètres plate-forme, système et machine.

plate-forme indique si vous utilisez une version Macintosh 68K, PowerPC ou Windows de 4e Dimension. Ce paramètre retourne une des constantes prédéfinies suivantes :

Constante	Type	Valeur
Macintosh 68K	Entier long	1
Power Macintosh	Entier long	2
Windows	Entier long	3

Les informations retournées dans système et machine dépendent de la version de 4e Dimension que vous utilisez.

Macintosh (versions 68K et PowerPC)

Avec une version MacOS de 4e Dimension, les paramètres système et machine retournent les informations suivantes :

• Le paramètre système retourne une valeur sur 32 bits (Entier long), dans laquelle le "mot machine haut" est inutilisé et le "mot machine bas" est structuré ainsi :

- L'octet supérieur contient le numéro de version principal,
- L'octet inférieur est composé de deux "nibbles" de 4 bits chacun. Le nibble supérieur est le numéro de mise à jour principal et le nibble inférieur le numéro de mise à jour secondaire. Par exemple : le système 9.0.4 est codé $0904, vous recevrez donc la valeur décimale 2308.

Note : Dans 4D, vous pouvez extraire ces valeurs à l'aide des Opérateurs numériques % (modulo) et \ (division entière) ou des Opérateurs sur les bits.

• Le paramètre machine retourne un numéro d'ID unique identifiant le modèle du Macintosh.

Note : Une liste à jour de ces numéros uniques est publiée par Apple Computer, Inc. dans sa documentation technique. De nouvelles valeurs apparaissent à mesure qu'Apple ou d'autres fabricants créent de nouveaux modèles de Macintosh.

Version Windows

Avec une version Windows de 4e Dimension, les paramètres système et machine retournent les informations suivantes :

• Le paramètre système retourne une valeur sur 32 bits (Entier long), structurée ainsi :

Si le bit supérieur vaut 0, vous utilisez Windows NT ou Windows 2000. S'il vaut 1, vous utilisez Windows 95 ou Windows 98.

Note : Le bit supérieur détermine le signe de la valeur Entier long. De ce fait, dans 4D, vous avez simplement besoin de tester la valeur ; si elle est positive, vous êtes sous Windows NT ou Windows 2000. Vous pouvez également utiliser les Opérateurs sur les bits.

L'octet inférieur fournit le numéro de version principal de Windows. S'il vaut 4, vous utilisez Windows 95, 98 ou Windows NT 4. S'il vaut 5, vous utilisez Windows 2000. Dans les deux cas, le signe de la valeur indique si vous utilisez Windows NT/2000 ou non.

L'octet inférieur suivant fournit le numéro de version secondaire de Windows. Sous Windows 98, sa valeur est 0A.

Note : Dans 4D, vous pouvez extraire ces valeurs à l'aide des Opérateurs numériques % (modulo) et \ (division entière) ou des Opérateurs sur les bits.

• Le paramètre machine retourne une des constantes prédéfinies suivantes :

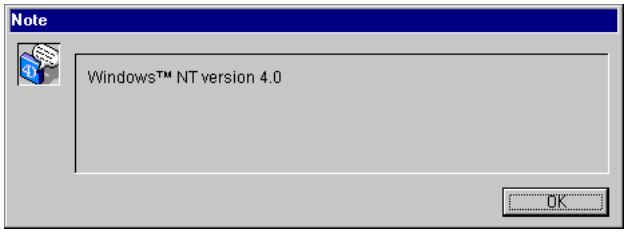
Constante	Type	Valeur
INTEL 386	Entier long	386
INTEL 486	Entier long	486
Pentium	Entier long	586
PowerPC 601	Entier long	601
PowerPC 603	Entier long	603
PowerPC 604	Entier long	604
PowerPC G3	Entier long	510
Autres G3 et supérieurs	Entier long	406

Exemple

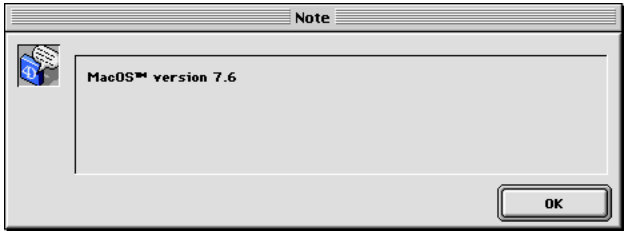
La méthode projet suivante affiche une boîte de dialogue d'alerte décrivant le système d'exploitation utilisé :

```
      ` Méthode projet VERSION SYSTEME

⇒     PROPRIETES PLATE FORME($vlPlatform;$vlSystem;$vlMachine)
      Si (($vlPlatform<1) | (3<$vlPlatform))
         $vsPlatformOS:=""
      Sinon
         Si ($vlPlatform=3)
            $vsPlatformOS:=""
            Si ($vlSystem<0)
               $winMajVers:=((2^31)+$vlSystem)%256
               $winMinVers:=(((2^31)+$vlSystem)\256)%256
               Si ($winMinVers=10)
                  $vsPlatformOS:="Windows™ 98"
               Sinon
                  $vsPlatformOS:="Windows™ 95"
               Fin de si
            Sinon
               $winMajVers:=$vlSystem%256
               $winMinVers:=($vlSystem\256)%256
               Si ($winMajVers=4)
                  $vsPlatformOS:="Windows™ NT"
               Sinon
                  $vsPlatformOS:="Windows™ 2000"
               Fin de si
            Fin de si
            $vsPlatformOS:=$vsPlatformOS+" version "+Chaine($winMajVers)+"."
                                                      +Chaine($winMinVers)
         Sinon
            $vsPlatformOS:="MacOS™ version "+Chaine($vlSystem\256)+"."
                              +Chaine(($vlSystem\16)%16)+(("."+Chaine($vlSystem%16))
                                                      *Num(($vlSystem%16) # 0))
         Fin de si
      Fin de si
      ALERTE($vsPlatformOS)
```

Sous Windows, vous obtenez une boîte de dialogue semblable à celle-ci :



Sous MacOS, vous obtenez une boîte de dialogue semblable à celle-ci :



Référence
Opérateurs sur les bits.

Fichier application → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Nom long du fichier 4D exécutable ou de l'application 4D

Description

La fonction Fichier application retourne le nom long (c'est-à-dire le chemin d'accès complet, y compris son nom) du fichier exécutable ou de l'application 4D que vous utilisez.

Sous Windows
Si, par exemple, vous utilisez 4e Dimension qui se trouve dans le répertoire \4DWIN600\PROGRAMME sur le volume E, Fichier application renvoie E:\4DWIN600\PROGRAMME\4D.EXE.

Sous MacOS
Si, par exemple, vous utilisez 4e Dimension qui se trouve dans le dossier 4e Dimension® 6.0*f* sur le disque Disque Dur, Fichier application renvoie Disque Dur:4e Dimension® 6.0*f*:4e Dimension® 6.0.

Exemple

Lorsque vous démarrez votre base sous Windows, vous souhaitez vérifier qu'une librairie DLL se trouve au même niveau que le fichier exécutable de 4D. Dans la Méthode base Sur ouverture, vous pouvez écrire les instructions suivantes :

```
    Si (Sous Windows & (Type application#4D Server))
⇒       Si (Tester chemin acces (Nom long vers chemin d'accès (Fichier application)+
                                                    "XRAYCAPT.DLL")#Est un document)
            ` Afficher une boîte de dialogue expliquant que la librairie XRAYCAPT.DLL
            ` n'est pas présente. Donc, la saisie de radios n'est pas disponible
        Fin de si
    Fin de si
```

Note : Les méthodes projet Sous Windows et Nom long vers chemin d'accès sont détaillées dans la section Présentation des documents système.

Fichier structure → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Nom long du fichier de structure de la base

Description

La fonction Fichier structure retourne le nom long (c'est-à-dire le chemin d'accès complet au fichier, y compris son nom) du fichier de structure de la base sur laquelle vous travaillez.

Sous Windows
Si, par exemple, vous travaillez sur la base MesCDs qui se trouve à \DOCS\MesCDs sur le volume G, Fichier structure renvoie G:\DOCS\MyCDs\MesCDs.4DB.

Sous Macintosh
Si, par exemple, vous travaillez sur la base MesCDs qui se trouve dans le dossier Documents:MesCDs*f*: sur le disque qui s'appelle Macintosh HD, Fichier structure renvoie Macintosh HD:Documents:MesCDs*f*:MesCDs.

ATTENTION : Si vous appelez cette commande lorsque vous utilisez 4D Client, seul le nom du fichier de structure est renvoyé, pas le nom long.

Exemple

Cet exemple affiche le nom et l'emplacement du fichier de structure que vous utilisez :

```
      Si(Type application#4D Client)
⇒        $vsStructNomFichier:=Nom long vers fichier(Fichier structure)
⇒        $vsStructNomChemin:=Nom long vers chemin d'accès (Fichier structure)
         ALERTE("Vous êtes en train d'utiliser la base "+Caractere(34)+$vsStructNomFichier
                                                +Caractere(34)+" qui se trouve au "+Caractere(34)
                                                      +$vsStructNomChemin+Caractere(34)+".")
      Sinon
⇒        ALERTE("Vous êtes connecté à la base "+Caractere(34)+Fichier structure
                                                                  +Caractere(34))
      Fin de si
```

Note : Les méthodes projet Nom long vers fichier et Nom long vers chemin d'accès sont détaillées dans la section Présentation des documents système.

Référence

Fichier application, Fichier donnees, LISTE SEGMENTS DE DONNEES.

Fichier donnees {(segment)} → Alpha

Paramètre	Type		Description
segment	Entier long	→	Numéro de segment
Résultat	Alpha	←	Nom long du fichier de données de la base

Description

La fonction Fichier donnees retourne le nom long (c'est-à-dire le chemin d'accès complet au fichier, y compris son nom) du fichier de données ou d'un segment de données de la base avec laquelle vous êtes en train de travailler.

Si vous ne passez pas le paramètre segment, Fichier donnees retourne le nom long du fichier de données ou du premier segment (si la base est segmentée). Si vous passez le paramètre segment, Fichier donnees retourne le nom long du segment de données qui correspond au numéro de segment passé. Si vous passez un numéro de segment supérieur au nombre de segments de données, Fichier donnees retourne une chaîne vide.

Sous Windows
Si, par exemple, vous travaillez avec la base MesCDs qui se trouve à l'emplacement \DOCS\MesCDs sur le volume G, Fichier donnees retournera G:\DOCS\MesCDs\MesCDs.4DD (si vous avez choisi l'emplacement proposé par défaut par 4D lorsque vous avez créé la base).

Sous MacOS
Si, par exemple, vous travaillez avec la base MesCDs qui se trouve dans le dossier Documents:MesCDs*f*: sur le disque Macintosh HD, Fichier donnees retournera Macintosh HD:Documents:MesCDs*f*:MesCDs.data (si vous avez choisi l'emplacement proposé par défaut par 4D lorsque vous avez créé la base).

ATTENTION : Si vous appelez cette fonction depuis 4D Client, seul le nom du fichier de données ou du premier segment de données est retourné, pas le nom long. De plus, même lorsque la base est segmentée, la commande retourne une chaîne vide pour les autres segments de données. Si vous voulez, pour des raisons d'administration, afficher la liste des segments de données sur un poste client, vous pouvez construire la liste et la stocker dans une variable sur le serveur à l'aide d'une procédure stockée. Ensuite, vous pouvez récupérer le contenu de cette variable en utilisant la commande LIRE VARIABLE PROCESS.

Exemple

Le code ci-dessous analyse les segments de données de la base :

```
Si(Type application#4D Client)
   $vlNumSegment:=0
   Repeter
      $vlNumSegment:=$vlNumSegment+1
⇒     $vsNomSegment:=Fichier donnees($vlNumSegment)
      Si ($vsNomSegment#"")
         ALERTE("Segment de données "+Chaine($vlNumSegment)+":"
                                    +Caractere(34)+$vsNomSegment+Caractere(34)+".")
      Fin de si
   Jusque ($vsNomSegment="")
   ALERTE("Il y a "+Chaine($vlNumSegment-1)+" segment(s) de données.")
Fin de si
```

Référence

Fichier application, Fichier structure, LISTE SEGMENTS DE DONNEES.

Dossier 4D → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Chemin d'accès au dossier 4D

Description

Dossier 4D renvoie le chemin d'accès au dossier 4D qui se trouve dans le dossier ou répertoire du système actif.

Sous Windows

Le dossier 4D se trouve dans le répertoire actif de WINDOWS (généralement C:\WINDOWS) et se nomme 4D. La fonction Dossier 4D retourne donc en général le chemin d'accès C:\WINDOWS\4D\. Cependant, les PC peuvent comporter des configurations multi-boot, et de plus l'emplacement et le nom du répertoire actif de WINDOWS peuvent être personnalisés au moment de l'installation. Par conséquent, si vous voulez sauvegarder vos propres fichiers ou documents dans le dossier 4D, cette commande vous permet d'obtenir avec certitude son chemin d'accès réel.

Sous MacOS

Le dossier 4D se trouve dans le dossier Préférences du dossier système actif et il se nomme 4D. En général, un chemin d'accès du type Macintosh HD:Système:Préférences:4D: sera donc retourné par la fonction Dossier 4D. Cependant, comme les utilisateurs Macintosh peuvent modifier le nom de leurs disques durs et de leur dossier système, le chemin d'accès peut être différent. Par conséquent, si vous voulez sauvegarder vos propres fichiers ou documents dans le dossier 4D, cette commande vous permet d'obtenir avec certitude son chemin d'accès réel.

Indépendance de plate-forme et international : En utilisant la fonction Dossier 4D pour obtenir le chemin d'accès à ce dossier, vous êtes certain que votre code fonctionnera correctement sur toute plate-forme, quelle que soit la langue du système.

L'environnement 4D utilise le dossier 4D pour stocker les informations suivantes :

- Fichiers d'identification utilisateur
- Fichiers de préférences utilisés par les applications, outils et programmes utilitaires de l'environnement 4D

• Composants réseaux 4D Client/4D Server ou Internet/Intranet (sous Windows, dans le sous-répertoire ...\4D\NETWORK) ainsi que leurs fichiers d'options
• les fichiers .rex et .res créés par 4D Client pour stocker les ressources téléchargées depuis 4D Server
• Les dossiers locaux des bases, créés par 4D Client pour stocker les extensions 4D téléchargées depuis 4D Server

ATTENTION : Vous pouvez stocker ce que vous voulez comme fichier ou document dans ce dossier, mais il est conseillé de ne pas modifier les fichiers de l'environnement 4D.

Exemple

Pendant le démarrage d'une base mono-utilisateur, vous voulez charger (ou créer) vos propres paramètres et les stocker dans un fichier situé dans le dossier 4D. Pour cela, dans la méthode base Sur ouverture, vous pouvez écrire les lignes suivantes :

```
ASSOCIER TYPES FICHIER("PREF";"PRF";"Préférences")
   ` Associer le type de fichier PREF sur MacOS à l'extension de fichier .PRF sur Windows
⇒  $vsNomDocPref:=Dossier 4D+"MesPrefs"
   ` Construire le chemin d'accès au fichier Préférences
Si(Tester chemin acces($vsNomDocPref+("PRF"#Num(Sous Windows)))#
                                                        Est un document)
      ` Vérifier si le fichier existe
   $vtRefDocPref:=Creer document($vsNomDocPref;"PREF") ` Si non, il faut le créer
Sinon
   $vtRefDocPref:=Ouvrir document($vsNomDocPref;"PREF") ` Si oui, il faut l'ouvrir
Fin de si
Si (OK=1)
   ` Traiter le contenu du document
   FERMER DOCUMENT($vtRefDocPref)
Sinon
      ` Gérer l'erreur
Fin de si
```

Référence

Dossier systeme, Dossier temporaire, Tester chemin acces.

LISTE SEGMENTS DE DONNEES (segments)

Paramètre	Type		Description
segments	Tableau alpha	←	Noms longs des segments de données de la base

Description

LISTE SEGMENTS DE DONNEES remplit le tableau segments avec les noms complets (chemin d'accès + nom de fichier) des segments de données de la base avec laquelle vous travaillez.

ATTENTION : Cette commande n'a pas d'effet si elle est exécutée sur 4D Client. Si vous avez besoin, pour des raisons d'administration, d'afficher une liste des segments de données sur un poste client, vous pouvez construire la liste et la stocker dans une variable sur le serveur à l'aide d'une procédure stockée. Ensuite, vous pouvez récupérer le contenu de cette variable en utilisant la commande LIRE VARIABLE PROCESS.

Exemples

(1) Dans le formulaire "Infos segments de données" de la table [Dialogues], vous voulez afficher une liste déroulante comportant les noms des segments de données. Pour ce faire, vous écrivez le code suivant :

```
   ` [Dialogues]; Méthode formulaire "Infos segments de données"
Au cas ou
   : (Evenement formulaire=Sur chargement)
      ` ...
      TABLEAU ALPHA(255;taNomSegment;0)
⇒     LISTE SEGMENTS DE DONNEES(taNomSegment)
      ` ...
Fin de cas
```

(2) La méthode suivante vous indique si votre base est segmentée :

```
   ` Fichier de données segmenté -> Booléen
C_BOOLEEN ($0)
⇒ LISTE SEGMENTS DE DONNEES($taNomSegment)
$0:=(Taille tableau($taNomSegment)>1)
```

(3) Après avoir appelé AJOUTER SEGMENT DE DONNEES, vous voulez tester si l'utilisateur a ajouté des segments de données.

```
⇒ LISTE SEGMENTS DE DONNEES($taAvant)
  AJOUTER SEGMENT DE DONNEES
⇒ LISTE SEGMENTS DE DONNEES($taAprès)
  Si(Taille tableau($taAvant)#Taille tableau($taAprès))
     ` Oui, un ou plusieurs segments de données ont été ajoutés
  Sinon
     ` Il y a le même nombre de segments de données
  Fin de si
```

Référence

Fichier application, Fichier donnees, Fichier structure.

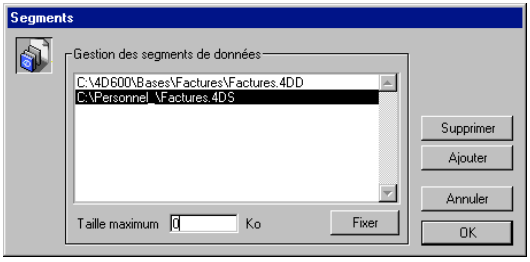
AJOUTER SEGMENT DE DONNEES

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande AJOUTER SEGMENT DE DONNEES provoque l'apparition de la boîte de dialogue de gestion des segments de données, présentée ci-dessous :



Si l'utilisateur clique sur le bouton **OK** pour valider la boîte de dialogue, la variable OK prend la valeur 1. Si l'utilisateur clique sur le bouton **Annuler**, OK prend la valeur 0.

NOTE : Cette commande ne fait rien lorsqu'elle est utilisée avec 4D Server.

Lorsque tous les segments de données sont pleins, 4e Dimension ou 4D Server génère une erreur –9999. Un message d'erreur est affiché, indiquant que le disque est plein.

Si vous utilisez 4e Dimension, vous pouvez intercepter l'erreur avec une méthode installée par APPELER SUR ERREUR. Vous pouvez alors appeler la commande AJOUTER SEGMENT DE DONNEES pour permettre à l'utilisateur de créer un nouveau segment sur un volume disposant d'espace libre. Si vous utilisez 4D Server, vous pouvez afficher une boîte de dialogue d'alerte indiquant que l'Administrateur de la base doit créer un nouveau segment sur le poste serveur.

Référence

APPELER SUR ERREUR.

Variables et ensembles système

OK vaut 1 si la boîte de dialogue de gestion des segments de données est validée.

ECRIRE CACHE

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande ECRIRE CACHE sauvegarde immédiatement les caches de données sur le disque. Toutes les modifications apportées à la base sont alors stockées sur disque.

Généralement, vous n'avez pas besoin d'appeler cette commande, car 4D sauvegarde régulièrement les modifications. Il est préférable d'utiliser la propriété de la base **Sauvegarde toutes les X minutes** (en mode Structure), qui spécifie les intervalles de sauvegarde des données, afin de contrôler l'écriture du cache de données sur le disque.

Note : 4D utilise en interne un système intégré de cache de données permettant d'accélérer les opérations d'E/S. Le fait que des modifications de données soient, par moments, présentes dans le cache de données et pas sur le disque est entièrement transparent pour votre code. Par exemple, si vous appelez la commande CHERCHER, le moteur de 4D va intégrer les données présentes dans le cache pour effectuer l'opération.

QUITTER 4D

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande QUITTER 4D vous permet de quitter 4e Dimension et de retourner sur le Bureau du système d'exploitation. Après un appel à QUITTER 4D, l'exécution du process courant est stoppée, puis 4D effectue les opérations suivantes :

• Si une Méthode base Sur fermeture existe, 4D l'exécute dans un nouveau process local. Par exemple, vous pouvez utiliser cette méthode base pour informer les autres process, via la communication interprocess, qu'ils doivent être fermés (s'ils sont en saisie de données) ou stopper l'exécution des opérations démarrées dans la Méthode base Sur ouverture (connexion de 4D à un autre serveur de bases de données). Notez que 4D quittera dans tous les cas : la Méthode base Sur fermeture peut assurer le nettoyage et la fermeture de toutes les opérations que vous voulez, mais la fermeture de la base est inéluctable.

• S'il n'existe pas de Méthode base Sur fermeture, 4D ferme tous les process un par un, sans distinction.

Si l'utilisateur est en saisie de données, les enregistrements seront annulés et non validés.

Si vous voulez permettre à l'utilisateur de sauvegarder ses modifications effectuées dans les fenêtres du process courant, vous pouvez utiliser la communication interprocess pour indiquer à tous les autres process utilisateur que la base est sur le point d'être quittée. Pour cela, vous pouvez adopter deux stratégies :

• Effectuer ces opérations depuis le process courant avant d'appeler QUITTER 4D.
• Traiter ces opérations depuis la Méthode base Sur fermeture.

Une troisième stratégie est également possible. Avant d'appeler QUITTER 4D, vous testez si une fenêtre nécessite une validation. Si c'est le cas, vous demandez à l'utilisateur de valider ou d'annuler cette fenêtre puis de choisir Quitter de nouveau. Cependant, du point de vue purement "interface utilisateur", les deux premières solutions sont préférables.

Exemple

La méthode projet suivante est associée à la commande **Quitter** du menu **Fichier**.

```
      ` Méthode projet M_QUITTER

   CONFIRMER("Etes-vous certain de vouloir quitter ?")
   Si (OK=1)
⇒     QUITTER 4D
   Fin de si
```

Référence

Méthode base Sur fermeture.

FIXER HISTORIQUE (historique | *)

Paramètre	Type		Description
historique | *	Alpha	→	Nom du fichier d'historique ou * pour refermer l'historique courant

Description

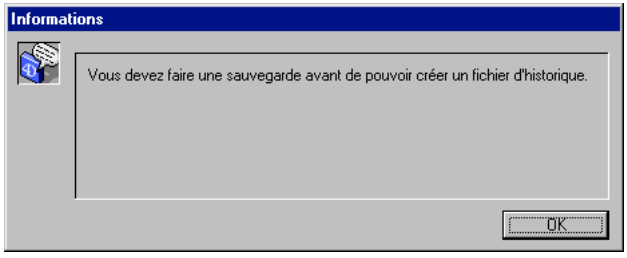
La commande FIXER HISTORIQUE ouvre, crée ou ferme le fichier d'historique de la base de données, suivant la valeur que vous passez dans historique.

IMPORTANT : Appeler la commande FIXER HISTORIQUE équivaut à choisir **Fichier d'historique...** dans le menu **Fichier** en mode Utilisation. Vous ne devez l'utiliser que si le plug-in 4D Backup est installé dans la base.

Si vous passez une chaîne vide dans historique, FIXER HISTORIQUE présente une boîte de dialogue standard d'ouverture de fichiers, permettant à l'utilisateur d'ouvrir un fichier d'historique ou d'en créer un nouveau. Si l'utilisateur clique sur le bouton **Ouvrir** et que le fichier est ouvert correctement, la variable OK prend la valeur 1. Autrement, si l'utilisateur clique sur le bouton **Annuler** ou si le fichier d'historique ne peut pas être ouvert ou créé, OK prend la valeur 0.

Si vous passez * dans historique, FIXER HISTORIQUE referme le fichier d'historique courant de la base. La variable OK prend la valeur 1 lorsque le fichier d'historique est refermé.

Si vous utilisez FIXER HISTORIQUE pour créer ou ouvrir un fichier d'historique avant qu'une sauvegarde intégrale n'ait été réalisée et si le fichier de données contient déjà des enregistrements, 4e Dimension affiche l'alerte suivante :



4D génère alors une erreur -4447, que vous pouvez intercepter avec une méthode installée par APPELER SUR ERREUR.

Note : La commande FIXER HISTORIQUE est sans effet lorsqu'elle est utilisée avec 4D Server. Pour plus d'informations sur cette commande, reportez-vous à la documentation du plug-in 4D Backup.

Référence

APPELER SUR ERREUR.

Variables et ensembles système

OK prend la valeur 1 si le fichier d'historique est correctement ouvert, créé ou fermé.

Gestion des erreurs

L'erreur -4447 est générée si l'opération ne peut pas être réalisée car la base de données doit être auparavant sauvegardée.

LIRE INFORMATIONS SERIALISATION

Environnement 4D

version 6.7

LIRE INFORMATIONS SERIALISATION (clé; utilisateur; société; connectés; maxUtilisateurs)

Paramètre	Type		Description
clé	Entier long	←	Clé unique du produit (crypté)
utilisateur	Alpha	←	Nom enregistré
société	Alpha	←	Organisation enregistrée
connectés	Entier long	←	Nombre d'utilisateurs connectés
maxUtilisateurs	Entier long	←	Nombre maximum d'utilisateurs

Description

La commande LIRE INFORMATIONS SERIALISATION retourne diverses informations relatives à la sérialisation de l'application 4D courante.

• clé : identifiant unique du produit installé. Ce numéro unique correspond à une seule application 4D (4D Server, 4e Dimension, 4D Runtime,etc.) installée sur un seul poste. Bien entendu, ce numéro est crypté.
• utilisateur : Nom de l'utilisateur de l'application, tel qu'il a été saisi au moment de l'installation.
• société : Nom de la société ou de l'organisation à laquelle appartient l'utilisateur, tel qu'il a été saisi au moment de l'installation.
• connectés : Nombre d'utilisateurs connectés au moment de l'exécution de la commande.
• maxUtilisateurs : Nombre maximal d'utilisateurs pouvant se connecter simultanément.

Note : Les deux derniers paramètres retournent toujours 1 pour les versions monopostes de 4D, sauf lorsqu'il s'agit de versions de démonstration, auquel cas ils retournent 0.

La commande LIRE INFORMATIONS SERIALISATION s'inscrit dans le schéma général de protection des "composants" proposé par 4D (pour plus d'informations sur les composants, reportez-vous à la documentation de 4D Insider).

Les développeurs de composants peuvent, s'ils le souhaitent, lier chaque copie de leur produit à une seule application 4D installée, afin d'empêcher toute copie illicite.

Le principe de fonctionnement du système est le suivant : un utilisateur souhaitant acquérir un composant fournit au développeur sa clé unique — générée à l'aide de la commande LIRE INFORMATIONS SERIALISATION. Cette opération peut, par exemple, être effectuée par l'intermédiaire d'un formulaire "Bon de commande" intégré à la version de démonstration du composant. La clé générée est unique : il n'existe qu'une clé par application 4D installée.

Le développeur du composant peut alors générer son propre numéro de série, en combinant la clé et l'algorithme de cryptage de son choix. Le composant livré comportera une fonction permettant de tester si les informations retournées par LIRE INFORMATIONS SERIALISATION correspondent bien à ce numéro de série. Dans le cas contraire, le composant sera rendu inutilisable.

Note : Les développeurs de plug-ins peuvent également bénéficier de ce système de protection. Pour plus d'informations, reportez-vous à la documentation de *4D External Kit*.

Référence

Lire ID ressource composant.

17

Environnement système

Hauteur ecran {(*)} → Numérique

Paramètre	Type		Description
*	*	→	Windows : hauteur de la fenêtre de l'application ou hauteur de l'écran si * est spécifié Macintosh : hauteur de l'écran principal
Résultat	Numérique	←	Hauteur exprimée en pixels

Description

Sous Windows, Hauteur ecran retourne la hauteur de la fenêtre d'application 4D (fenêtre MDI). Si vous passez le paramètre optionnel *, Hauteur ecran retourne la hauteur de l'écran.

Sous MacOS, Hauteur ecran retourne la hauteur de l'écran principal, c'est-à-dire celui qui contient la barre de menus.

Référence

COORDONNEES ECRAN, Largeur ecran.

Largeur ecran {(*)} → Numérique

Paramètre	Type		Description
*	*	→	Windows : largeur de la fenêtre de l'application ou largeur de l'écran si * est spécifié Macintosh : largeur de l'écran principal
Résultat	Numérique	←	Largeur exprimée en pixels

Description

Sous Windows, Largeur ecran retourne la largeur de la fenêtre d'application 4D (fenêtre MDI). Si vous passez le paramètre optionnel *, Largeur ecran retourne la largeur de l'écran.

Sous MacOS, Largeur ecran retourne la largeur de l'écran principal, c'est-à-dire celui qui contient la barre de menus.

Référence

COORDONNEES ECRAN, Hauteur ecran.

Nombre ecrans → Entier long

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	Nombre d'écrans

Description

Nombre ecrans renvoie le nombre de moniteurs qui sont connectés à votre machine.

Note pour les utilisateurs Windows : Sous Windows, Nombre ecrans renvoie toujours 1.

Référence

COORDONNEES ECRAN, Ecran principal, Hauteur ecran, Largeur ecran, PROFONDEUR ECRAN.

COORDONNEES ECRAN (gauche; haut; droite; bas{; écran})

Paramètre	Type		Description
gauche	Entier long	←	Coordonnée gauche de la zone de l'écran
haut	Entier long	←	Coordonnée supérieure de la zone de l'écran
droite	Entier long	←	Coordonnée droite de la zone de l'écran
bas	Entier long	←	Coordonnée inférieure de la zone de l'écran
écran	Entier long	→	Numéro de l'écran ou écran principal si omis

Description

La commande COORDONNEES ECRAN retourne dans les paramètres gauche, haut, droite et bas les coordonnées de l'écran spécifié dans le paramètre écran.

Sous Windows
Généralement, vous n'aurez pas besoin de passer le paramètre écran.

Sous Macintosh
Si vous omettez le paramètre écran, COORDONNEES ECRAN retourne les coordonnées de l'écran principal, c'est-à-dire celui dans lequel la barre de menus est affichée.

Référence

Ecran principal, Nombre ecrans, PROFONDEUR ECRAN.

PROFONDEUR ECRAN (profondeur; couleurs{; écran})

Paramètre	Type		Description
profondeur	Numérique	←	Profondeur de l'écran (nombre de couleurs = 2 ^ profondeur)
couleurs	Numérique	←	1 = écran couleur 0 = écran noir et blanc ou niveaux de gris
écran	Numérique	→	Numéro de l'écran ou écran principal si omis

Description

La commande PROFONDEUR ECRAN retourne dans les paramètres profondeur et couleurs les caractéristiques du moniteur utilisé.

Après l'appel :

• La profondeur de l'écran est retournée dans profondeur. La profondeur élevée en tant que puissance de 2 vous permet de connaître le nombre de couleurs que votre moniteur affiche. Si par exemple votre moniteur est paramétré en 256 couleurs (2^8), la profondeur de votre écran est de 8.

4e Dimension fournit les constantes prédéfinies suivantes :

Constante	Type	Valeur
Noir et blanc	Entier long	0
Quatre couleurs	Entier long	2
Seize couleurs	Entier long	4
Deux cent cinquante six coul	Entier long	8
Milliers de couleurs	Entier long	16
Million de couleurs 24 bits	Entier long	24
Million de couleurs 32 bits	Entier long	32

Si le moniteur est configuré pour afficher des couleurs, le paramètre couleur vaut 1. Si le moniteur est configuré pour afficher des niveaux de gris, couleur vaut 0 (zéro). Notez que cette valeur n'a de signification que sous MacOS.

Vous disposez des constantes prédéfinies suivantes :

Constante	Type	Valeur
Est en niveaux de gris	Entier long	0
Est en couleurs	Entier long	1

• Le paramètre optionnel écran vous permet de spécifier le numéro du moniteur sur lequel vous souhaitez obtenir des informations. Sous Windows, vous n'aurez généralement pas besoin d'utiliser ce paramètre. Sous MacOS, si vous omettez le paramètre écran, la commande retourne la profondeur de l'écran principal, c'est-à-dire celui qui contient la barre de menus.

Exemple

Votre application affiche de nombreux graphiques en couleurs. Vous pouvez écrire, quelque part dans votre base :

```
⇒    PROFONDEUR ECRAN ($vlProf;$vlCouleur)
     Si ($vlProf<8)
        ALERTE("Les formulaires seraient plus beaux si le moniteur"+" était configuré
                                                          en 256 couleurs ou plus.")
     Fin de si
```

Référence

FIXER PROFONDEUR ECRAN, Nombre ecrans.

FIXER PROFONDEUR ECRAN

Environnement système

version 6.0

FIXER PROFONDEUR ECRAN (profondeur{; couleurs{; écran}})

Paramètre	Type		Description
profondeur	Numérique	→	Profondeur de l'écran (nombre de couleurs = 2 ^ profondeur)
couleurs	Numérique	→	1 = écran couleur 0 = écran en niveaux de gris
écran	Numérique	→	Numéro de l'écran ou écran principal si omis

Description

Sous Windows, cette commande ne fait rien.

Sous MacOS, FIXER PROFONDEUR ECRAN vous permet de modifier la profondeur et les paramètres de couleurs/niveaux de gris de l'écran dont vous avez passé le numéro dans écran. Si vous ne passez pas ce paramètre, la commande s'applique à l'écran principal.

Pour connaître les valeurs à passer dans les paramètres profondeur et couleurs, reportez-vous à la description de la commande PROFONDEUR ECRAN.

Référence

PROFONDEUR ECRAN.

Ecran principal → Entier long

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	Numéro de l'écran contenant la barre de menus

Description

La commande Ecran principal retourne le numéro de l'écran dans lequel se trouve la barre de menus.

Note pour les utilisateurs Windows : Sous Windows, Ecran principal renvoie toujours 1.

Référence

Hauteur barre de menus, Nombre ecrans.

Hauteur barre de menus → Entier long

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	Hauteur (exprimée en pixels) de la barre de menus (retourne zéro si la barre de menus est cachée)

Description

La commande Hauteur barre de menus retourne la hauteur de la barre de menus, exprimée en pixels.

Référence

AFFICHER BARRE DE MENUS, CACHER BARRE DE MENUS, Ecran principal.

LISTE DES POLICES (polices)

Paramètre	Type		Description
polices	Tableau	←	Tableau des noms des polices disponibles

Description

La commande LISTE DES POLICES remplit le tableau polices (de type Alpha ou Texte) avec les noms des polices disponibles dans votre système.

Exemple

Dans un formulaire, vous voulez obtenir une liste déroulante qui affiche les polices disponibles dans le système. Ecrivez la méthode suivante pour votre objet liste déroulante :

```
   Au cas ou
      : (Evenement formulaire=Sur chargement)
         TABLEAU ALPHA (63;taPolices;0)
⇒        LISTE DES POLICES(taPolices)
         ` ...
   Fin de cas
```

Référence

Nom de police, Numero de police.

Nom de police (numéro) → Alpha

Paramètre	Type		Description
numéro	Entier long	→	Numéro de la police de laquelle récupérer le nom
Résultat	Alpha	←	Nom de la police

Description

La fonction Nom de police retourne le nom de la police dont le numéro est passé dans le paramètre numéro. Si aucune police de ce numéro n'est disponible, Nom de police retourne une chaîne vide.

Exemples

(1) Pour afficher un objet dans votre formulaire avec la police du système par défaut, écrivez la ligne suivante :

```
` 0 est le numéro de la police du système
⇒ CHANGER JEU DE CARACTERES(monObjet;Nom de police(0))
```

(2) Pour afficher un objet dans votre formulaire avec la police de l'application par défaut, écrivez la ligne suivante :

```
` 1 est le numéro de la police de l'application
⇒ CHANGER JEU DE CARACTERES(monObjet;Nom de police(1))
```

Référence

LISTE DES POLICES, Numero de police.

Numero de police (nomPolice) → Entier long

Paramètre	Type		Description
nomPolice	Alpha	→	Nom de la police de laquelle récupérer le numéro
Résultat	Entier long	←	Numéro de police

Description

La fonction Numero de police retourne le numéro de la police dont le nom est passé dans le paramètre nomPolice. Si aucune police de ce nom n'existe, Numero de police retourne 0.

Exemple

Si certains formulaires dans votre base utilisent une police qui s'appelle “Spéciale”, vous pouvez écrire le code suivant :

```
⇒   Si (Numero de police("Spéciale")=0)
       ALERTE("Vous devriez installer la police Spéciale pour que ce formulaire soit
                                                                  correctement affiché.")
    Fin de si
```

Référence

LISTE DES POLICES, Nom de police.

Dossier systeme

Environnement système

version 6.7 (Modifiée)

Dossier systeme {(type)} → Alpha

Paramètre	Type		Description
type	Entier long	→	Type de dossier système
Résultat	Alpha	←	Chemin d'accès d'un dossier du système actif

Description

La fonction Dossier systeme retourne le chemin d'accès du dossier Système Windows ou MacOS actif, ou le chemin d'accès d'un dossier particulier à l'intérieur du dossier système actif.

Le paramètre optionnel type vous permet d'indiquer le type de dossier dont vous souhaitez obtenir le chenim d'accès. Si vous ne passez pas ce paramètre, Dossier systeme retourne le chemin d'accès du dossier Système Windows ou MacOS actif.
Vous passez dans type un code représentant le type de dossier. 4D fournit les constantes prédéfinies suivantes (placées dans le thème "Documents système") :

Constante	Type	Valeur
Système	Entier long	0
Polices	Entier long	1
Préférences_Tous	Entier long	2
Préférences	Entier long	3
Ouverture au démarrage_Tous	Entier long	4
Ouverture au démarrage	Entier long	5
Ouverture à l'extinction_Tous	Entier long	6
Ouverture à l'extinction	Entier long	7
Menu Pomme ou Démarrer_Tous	Entier long	8
Menu Pomme ou Démarrer	Entier long	9
Extensions Mac	Entier long	10
Tableaux de bord Mac	Entier long	11

• Les deux dernières constantes (Extensions Mac et Tableaux de bord Mac) sont réservées à une utilisation sous MacOS. Lorsqu'elles sont utilisées sous Windows, Dossier systeme retourne une chaîne vide.

• L’emplacement de certains dossiers peut être différent suivant le type de session ouverte par l’utilisateur. Les constantes 2 à 9 vous permettent de choisir si vous souhaitez obtenir le chemin d’accès du dossier spécifique à l’utilisateur courant (constantes simples) ou commun à tous les utilisateurs (constantes suivies de “Tous”).

Note : Sous Windows, cette commande nécessite la présence du fichier système “SHFolder.DLL” **version 4.71 minimum** dans le dossier système actif de Windows. Sinon, la commande retourne des chaînes vides pour tous les types de dossiers, hormis pour le type 0 (Système).

Référence

Dossier 4D, Dossier temporaire.

Dossier temporaire → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Chemin d'accès au dossier temporaire

Description

La fonction Dossier temporaire retourne le chemin d'accès au dossier temporaire courant tel que défini par votre système d'exploitation.

Exemple

Reportez-vous à l'exemple de la commande AJOUTER A PRESSE PAPIERS.

Référence

Dossier systeme.

Nom de la machine → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Nom de la machine sur le réseau

Description

La commande Nom de la machine retourne le nom de votre machine tel qu'il a été défini dans le tableau de bord Réseau.

Exemple

Même si vous n'utilisez pas la version client/serveur de 4D, votre application peut comprendre des services réseaux qui nécessitent que votre machine soit correctement configurée. Dans la Méthode base Sur ouverture de votre application, vous pouvez écrire :

```
⇒ Si ( (Nom de la machine="") | (Nom du possesseur="") )
     ` Afficher ici une boîte de dialogue demandant à l'utilisateur de configurer
     ` ses paramètres réseau
  Fin de si
```

Référence

Nom du possesseur.

Nom du possesseur → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Nom du possesseur de la machine sur le réseau

Description

La fonction Nom du possesseur retourne le nom du possesseur de la machine, tel qu'il a été défini dans le tableau de bord Réseau.

Exemple

Reportez-vous à l'exemple de la commande Nom de la machine.

Référence

Nom de la machine.

Gestalt (sélecteur; valeur) → Numérique

Paramètre	Type		Description
sélecteur	Alpha	→	Sélecteur gestalt (4 caractères)
valeur	Numérique	←	Résultat du gestalt
Résultat	Numérique	←	Code d'erreur résultant

Description

La commande Gestalt retourne dans valeur une valeur numérique représentant les caractéristiques de la configuration matérielle et logicielle de votre système, en fonction du sélecteur que vous avez passé dans le paramètre sélecteur.

Si l'information demandée est obtenue, la fonction Gestalt retourne 0, sinon elle retourne l'erreur -5550. Si le sélecteur est inconnu, Gestalt retourne l'erreur -5551.

Important : Le Gestalt Manager est spécifique à MacOS. Certains des sélecteurs sont également implémentés sous Windows mais l'utilité de cette fonction reste limitée sur cette plate-forme.

Pour plus d'informations sur les sélecteurs que vous pouvez passer dans Gestalt, reportez-vous à la documentation technique Apple relative au Gestalt Manager, consultable en ligne à l'adresse suivante : http://til.info.apple.com/techinfo.nsf/artnum/n9095.

Exemple

Dans la version 7.6 de MacOS, le code suivant :

```
⇒   $vlErrCode:=Gestalt("sysv";$vlInfo)
    Si ($vlErrCode=0)
       ALERTE("Vous utilisez la version suivante du système : "+Chaîne($vlInfo;"&x"))
    Fin de si
```

... affiche l'alerte “Vous utilisez la version suivante du système : 0x0760”.

ENREGISTRER EVENEMENT (message{; importance})

Paramètre	Type		Description
message	Alpha	→	Contenu du message
importance	Entier	→	Niveau d'importance du message

Note : Cette fonctionnalité est disponible sous Windows NT uniquement.

Description

4D permet de tirer parti de l'"Observateur d'événements" de Windows NT. Ce journal reçoit et stocke des messages en provenance des applications en cours d'exécution. Il permet donc de contrôler le déroulement d'une session de travail. Pour plus d'informations sur ce point, reportez-vous au manuel *Mode Structure*.
Pour que cette fonctionnalité soit disponible, le service Journal d'événements de Windows NT doit être démarré.

La commande ENREGISTRER EVENEMENT vous permet d'ajouter des messages personnalisés qui apparaîtront dans l'Observateur d'événements de Windows NT.

Vous passez dans message le message à inscrire dans le journal d'événements.

Vous pouvez attribuer à ce message un niveau d'importance, ce qui facilite la lecture du journal d'événements. Il existe trois niveaux d'importance : Information, Avertissement et Erreur. Le paramètre importance vous permet de fixer le niveau d'importance de message.
4e Dimension vous propose les constantes prédéfinies suivantes, placées dans le thème "Journal d'événements Windows NT" :

Constante	Type	Valeur
Message d'information (valeur par défaut)	Entier	0
Message d'avertissement	Entier	1
Message d'erreur	Entier	2

Si vous ne passez pas le paramètre importance ou passez une valeur invalide, la valeur par défaut (0) est utilisée.

Exemple

Afin de conserver une trace des lancements de votre base, vous pouvez écrire, dans la Méthode base Sur ouverture :

```
⇒ ENREGISTRER EVENEMENT ("Démarrage de la base Facturation")
```

A chaque ouverture de la base, cette information sera inscrite dans l'Observateur d'événements de Windows NT, avec le niveau d'importance 0.

18

Evénements formulaire

Evenement formulaire → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Numéro d'événement formulaire

Description

Evenement formulaire retourne une valeur numérique qui identifie le type d'événement formulaire qui vient de se produire. Généralement, Evenement formulaire s'utilise dans une méthode formulaire ou une méthode objet.

4e Dimension fournit les constantes prédéfinies suivantes :

Constante	Valeur	Description
Sur chargement	1	Le formulaire s'affiche ou s'imprime
Sur libération	24	Le formulaire se referme et est déchargé
Sur validation	3	La saisie des données dans l'enregistrement est validée
Sur clic souris	4	Un clic est survenu sur un objet
Sur double clic souris	13	Un double-clic est survenu sur un objet
Sur avant frappe clavier	17	Un caractère vient d'être saisi dans l'objet qui a le focus Lire texte edite retourne le contenu sans ce caractère
Sur après frappe clavier	28	Un caractère vient d'être saisi dans l'objet qui a le focus Lire texte edite retourne le contenu avec ce caractère
Sur gain focus	15	Un objet de formulaire va avoir le focus
Sur perte focus	14	Un objet de formulaire perd le focus
Sur activation	11	La fenêtre du formulaire passe au premier plan
Sur désactivation	12	La fenêtre du formulaire passe en arrière-plan
Sur appel extérieur	10	Le formulaire a reçu un appel de la commande APPELER PROCESS
Sur déposer	16	Des données sont déposées sur un objet
Sur glisser	21	Des données peuvent être déposées sur un objet
Sur menu sélectionné	18	Une commande de menu a été sélectionnée
Sur données modifiées	20	Les données d'un objet ont été modifiées
Sur appel zone du plug in	19	Un plug-in demande que sa méthode objet soit exécutée

Sur entête	5	L'en-tête du formulaire va être imprimé ou affiché
Sur impression corps	23	Le corps du formulaire va être imprimé
Sur impression sous total	6	Une rupture du formulaire va être imprimée
Sur impression pied de page	7	Le pied de page du formulaire va être imprimé
Sur case de fermeture	22	On a cliqué sur la case de fermeture de la fenêtre
Sur affichage corps	8	Un enregistrement va être affiché dans la liste
Sur ouverture corps	25	On a double-cliqué sur un enregistrement et on passe au formulaire entrée
Sur fermeture corps	26	Le formulaire entrée se referme et on retourne au formulaire sortie
Sur minuteur	27	Le nombre de ticks défini par FIXER MINUTEUR est atteint
Sur redimensionnement	29	La fenêtre du formulaire est redimensionnée

Evénements et méthodes

Lorsqu'un événement formulaire se produit, 4e Dimension effectue les actions suivantes :

• En premier lieu, il examine chaque objet du formulaire et appelle la méthode de ceux dont la propriété d'événement correspondante a été sélectionnée et qui sont impliqués dans l'événement.

• Ensuite, il appelle la méthode formulaire si la propriété d'événement correspondante a été sélectionnée pour le formulaire.

Les différentes méthodes objet ne sont pas appelées dans un ordre particulier. La règle est que les méthodes objet sont toujours appelées avant la méthode formulaire. Dans le cas des sous-formulaires, les méthodes objet du formulaire sortie du sous-formulaire sont d'abord appelées, puis la méthode formulaire du formulaire sortie, puis enfin 4D appelle les méthodes objet du formulaire parent. Autrement dit, lorsqu'un objet est un sous-formulaire, 4D utilise la même règle pour les méthodes formulaire et objet dans le sous-formulaire.

Lorsque, pour un événement particulier, la propriété d'événement du formulaire n'est pas sélectionnée, cela n'empêche pas les appels aux méthodes des objets pour lesquels l'événement est sélectionné. Autrement dit, la sélection ou la désélection d'un événement au niveau du formulaire n'a pas d'effet sur les propriétés d'événements des objets.

ATTENTION : Ce principe ne s'applique pas aux événements Sur chargement et Sur libération. Ces événements ne seront générés pour un objet que si les propriétés d'événement correspondantes ont été sélectionnées à la fois pour l'objet et pour le formulaire auquel il appartient. Si les propriétés sont sélectionnées pour l'objet uniquement, les événements ne seront pas générés ; ces deux événements doivent être sélectionnés au niveau du formulaire.

Le nombre d'objets impliqués par un événement dépend de la nature de l'événement. En particulier :

• Pour l'événement Sur chargement, les méthodes objet de tous les objets du formulaire (sur toutes les pages) pour lequels la propriété d'événement Sur chargement est sélectionnée seront appelées. Si l'événement Sur chargement est sélectionné pour le formulaire, la méthode formulaire sera appelée.

• Pour les événements Sur activation ou Sur redimensionnement, aucune méthode objet ne sera appelée car ces événements s'appliquent au formulaire, pas à un objet en particulier. Par conséquent, si ces événements sont sélectionnés pour le formulaire, seule la méthode formulaire sera appelée.

• Pour l'événement Sur glisser, seule la méthode de l'objet déposable impliqué par l'événement sera appelée (si, bien entendu, la propriété d'événement Sur glisser est sélectionnée pour l'objet). La méthode formulaire ne sera pas appelée.

• Les événements Sur ouverture corps et Sur fermeture corps ne fonctionnent que dans le contexte d'un formulaire sortie affiché par la commande VISUALISER SELECTION ou MODIFIER SELECTION. Ils doivent être placés dans la méthode de ce formulaire.

• L'événement Sur minuteur n'est généré que si la méthode formulaire contient un appel préalable à la commande FIXER MINUTEUR. Seule la méthode formulaire reçoit cet événement, aucune méthode objet ne sera appelée.

ATTENTION : Contrairement à tous les autres événements, la méthode d'un objet, pendant l'événement Sur glisser, est exécutée dans le contexte du process de l'objet source du glisser-déposer, et non dans celui du process de l'objet de destination de l'opération. Pour plus d'informations, reportez-vous aux descriptions des commandes PROPRIETES GLISSER DEPOSER et Position deposer.

Le tableau suivant résume, pour chaque type d'événement, l'appel des méthodes formulaire et objet :

Evénement	Méthode(s) objet	Méthode formulaire	Quel(s) objet(s)
Sur chargement	Oui	Oui	Tous
Sur libération	Oui	Oui	Tous
Sur validation	Oui	Oui	Tous
Sur clic souris	Oui (si cliquable) (*)	Oui	Seul l'objet impliqué
Sur double clic souris	Oui (si cliquable) (*)	Oui	Seul l'objet impliqué
Sur avant frappe clavier	Oui (si saisissable) (*)	Oui	Seul l'objet impliqué
Sur après frappe clavier	Oui (si saisissable) (*)	Oui	Seul l'objet impliqué
Sur gain focus	Oui (si tabulable) (*)	Oui	Seul l'objet impliqué
Sur perte focus	Oui (si tabulable) (*)	Oui	Seul l'objet impliqué
Sur activation	Jamais	Oui	Aucun
Sur désactivation	Jamais	Oui	Aucun

Sur appel extérieur	Jamais	Oui	Aucun
Sur déposer	Oui (si déposable) (*)	Oui	Seul l'objet impliqué
Sur glisser	Oui (si déposable) (*)	Jamais	Seul l'objet impliqué
Sur menu sélectionné	Jamais	Oui	Aucun
Sur données modifiées	Oui (si modifiable) (*)	Oui	Seul l'objet impliqué
Sur appel zone du plug in	Oui	Oui	Seul l'objet impliqué
Sur entête	Oui	Oui	Tous
Sur impression corps	Oui	Oui	Tous
Sur impression sous total	Oui	Oui	Tous
Sur impression pied de page	Oui	Oui	Tous
Sur case de fermeture	Jamais	Oui	Aucun
Sur affichage corps	Oui	Oui	Tous
Sur ouverture corps	Jamais	Oui	Aucun
Sur fermeture corps	Jamais	Oui	Aucun
Sur redimensionnement	Jamais	Oui	Aucun
Sur minuteur	Jamais	Oui	Aucun

(*) Référez-vous ci-dessous au paragraphe "Evénements, objets et propriétés" pour plus d'informations.

IMPORTANT : Gardez constamment à l'esprit que, pour chaque événement, la méthode d'un formulaire ou d'un objet est appelée si l'événement correspondant a été sélectionné pour le formulaire ou l'objet. L'avantage de désactiver des événements en mode Structure (en utilisant les fenêtres de propriétés d'objet et de formulaire) est que vous pouvez réduire de manière très importante le nombre d'appels aux méthodes et donc optimiser la vitesse d'exécution des formulaires.

Evénements, objets et propriétés

Une méthode objet est appelée si l'événement peut réellement se produire pour l'objet en fonction de sa nature et de ses propriétés. Ce paragraphe détaille les événements à utiliser pour gérer les différents types d'objets.

Objets cliquables

Les objets cliquables sont gérés principalement avec la souris. Ces objets sont les suivants :

- Variables ou champs saisissables de type Booléen
- Boutons, boutons par défaut, boutons radio, cases à cocher, grilles de boutons
- Boutons 3D, boutons radio 3D, cases à cocher 3D
- Pop up menus, pop up menus hiérarchiques, menus Images
- Listes déroulantes, menus,
- Zones de défilement, listes hiérarchiques

• Boutons invisibles, boutons inversés, boutons radio image
• Thermomètres, règles, cadrans
• Onglets,
• Séparateurs.

Lorsque l'événement Sur clic souris ou Sur double clic souris est sélectionné pour un de ces objets, vous pouvez détecter et gérer les clics sur l'objet à l'aide de la la commande Evenement formulaire qui retourne Sur clic souris ou Sur double clic souris selon le cas.
Si les deux événements sont sélectionnés pour un même objet, les événements Sur clic souris puis Sur double clic souris seront générés en cas de double-clic sur l'objet.

Pour tous les objets cliquables, l'événement Sur clic souris se produit une fois que le bouton de la souris est relâché. Il y a cependant deux exceptions :
• Avec des boutons invisibles, l'événement Sur clic souris se produit dès qu'un clic a été détecté — sans attendre le relâchement du bouton de la souris.
• Avec les thermomètres, règles et cadrans, si le format d'affichage indique que la méthode objet doit être appelée pendant que vous faites glisser les curseurs de contrôle, l'événement Sur clic souris survient dès que le clic est détecté.

Note : Quelques objets peuvent être activés par le clavier. Une case à cocher, par exemple, une fois qu'elle a le focus, peut être sélectionnée à l'aide de la barre d'espace. Dans ce cas, l'événement Sur clic souris est quand même généré.

ATTENTION : Les combo-boxes ne sont pas considérées comme des objets cliquables. Une combo-box doit être perçue comme une zone de texte saisissable dont la liste déroulante fournit les valeurs par défaut. Par conséquent, vous gérez la saisie des données dans une combo-box à l'aide des événements Sur avant frappe clavier, Sur après frappe clavier et Sur données modifiées.

Objets saisissables par clavier

Les objets saisissables par clavier sont des objets dans lesquels vous saisissez des données par le clavier et pour lesquels vous pouvez filtrer les données au plus bas niveau en détectant les événements Sur avant frappe clavier et Sur après frappe clavier. Vous pouvez tirer pleinement parti de ces événements à l'aide de la commande Lire texte edite.
Les objets saisissables sont les suivants :

• Tous les champs saisissables (sauf images, sous-formulaires et BLOB)
• Toutes les variables saisissables (sauf images, BLOB, pointeurs et tableaux)
• Combo-boxes

Une fois que les événements Sur avant frappe clavier et Sur après frappe clavier ont été sélectionnés pour un de ces objets, vous pouvez détecter et gérer la saisie par le clavier dans l'objet à l'aide de la commande Evenement formulaire qui va retourner Sur avant frappe clavier puis Sur après frappe clavier (pour plus d'informations, reportez-vous à la description de la commande Lire texte edite).

Notes :
• La commande GENERER FRAPPE CLAVIER active les événements formulaire Sur avant frappe clavier et Sur après frappe clavier.
• Bien qu'objets "saisissables", les listes hiérarchiques ne génèrent pas les événements formulaire Sur avant frappe clavier et Sur après frappe clavier.

Objets modifiables

Les objets modifiables sont des objets ayant une source de données, dont la valeur peut être modifiée à l'aide de la souris ou du clavier, mais qui ne sont pas gérés par l'événement Sur clic souris. Ces objets sont les suivants :

• Tous les champs saisissables (sauf sous-tables et BLOB)
• Toutes les variables saisissables (sauf BLOB, pointeurs et tableaux)
• Combo-boxes
• Objets externes (pour lesquels la saisie de données est validée par le plug-in 4D)

Ces objets reçoivent l'événement Sur données modifiées. Lorsque l'événement Sur données modifiées est sélectionné pour un de ces objets, vous pouvez détecter et gérer la modification des valeurs à l'aide de la commande Evenement formulaire qui retourne Sur données modifiées.

Objets tabulables

Les objets tabulables sont ceux qui peuvent recevoir le focus lorsque vous utilisez la touche **Tab** et/ou si vous cliquez dessus. L'objet qui a le focus est celui qui reçoit les caractères saisis via le clavier et qui ne sont pas les accélerateurs (Windows) ou les raccourcis-clavier (MacOS) d'une commande de menu ou d'un objet tel qu'un bouton.

TOUS les objets sont tabulables SAUF ceux listés ci-dessous :

• Variables ou champs non saisissables
• Boutons (sous MacOS)
• Grilles de boutons
• Boutons 3D, boutons radio 3D, cases à cocher 3D
• Pop-up/listes déroulantes (sous MacOS)
• Menus déroulants hiérarchiques
• Pop-up menus Image
• Zones de défilement
• Boutons invisibles, boutons inversés, boutons radio Images
• Graphes
• Objets externes (pour lesquels la saisie de données est validée par le plug-in 4D)

• Onglets
• Séparateurs

Lorsque les événements Sur gain focus et/ou Sur perte focus sont sélectionnés pour un objet tabulable, vous pouvez détecter et gérer la modification du focus à l'aide de la commande Evenement formulaire qui retournera Sur gain focus ou Sur perte focus en fonction de l'événement.

Catégories d'événements

Les événements formulaires peuvent être classés dans les catégories suivantes :

• Evénements généraux :
Sur chargement, Sur libération, Sur validation, Sur affichage corps, Sur ouverture corps, Sur fermeture corps
• Evénements propres aux formulaires :
Sur activation, Sur désactivation, Sur appel extérieur, Sur case de fermeture, Sur menu sélectionné, Sur minuteur, Sur redimensionnement
• Evénements liés aux actions de l'utilisateur :
Sur clic souris, Sur double-clic souris, Sur avant frappe clavier, Sur après frappe clavier, Sur gain focus, Sur perte focus, Sur données modifiées, Sur appel zone du plug in
• Evénements liés au glisser-déposer :
Sur déposer, Sur glisser
• Evénements liés à l'impression :
Sur entête, Sur impression corps, Sur impression sous total, Sur impression pied de page

Le tableau ci-dessous résume l'équivalence entre les événements formulaire (4D V6) et les cycles d'exécution des formulaires (4D V5).

Evénements V6	Cycles d'exécution du formulaire V5	Commande V5
Sur chargement	Phase avant	Avant
Sur libération	Pas d'équivalent	Aucune
Sur validation	Phase après	Apres
Sur clic souris	Phase pendant générique	Pendant
Sur double clic souris	Phase pendant générique	Pendant
Sur avant frappe clavier	Pas d'équivalent	Aucune
Sur après frappe clavier	Pas d'équivalent	Aucune
Sur gain focus	Pas d'équivalent	Aucune
Sur perte focus	Pas d'équivalent	Aucune
Sur activation	Phase activation	Activation
Sur désactivation	Phase désactivation	Desactivation
Sur appel extérieur	Phase appel extérieur	Appel exterieur
Sur déposer	Pas d'équivalent	Aucune
Sur glisser	Pas d'équivalent	Aucune
Sur menu sélectionné	Phase pendant générique	Pendant plus Menu choisi
Sur données modifiées	Phase pendant générique	Pendant
Sur appel zone du plug in	Phase pendant générique	Pendant
Sur entête	Phase d'impression d'en-tête	En entete
Sur impression corps	Phase pendant générique	Pendant
Sur impression sous total	Phase d'impression de sous-total	En rupture
Sur impression pied de page	Phase d'impression de pied de page	En pied
Sur case de fermeture	Pas d'équivalent	Creer fenetre (avec case de fermeture)
Sur affichage corps	Phase avant et pendant	Avant & Pendant
Sur ouverture corps	Phase pendant générique	Pendant
Sur fermeture corps	Phase pendant générique	Pendant
Sur minuteur	Pas d'équivalent	Aucune
Sur redimensionnement	Pas d'équivalent	Aucune

Lorsque vous ouvrez une base V5 avec 4e Dimension V6, le programme effectue deux opérations :

- 4D convertit le fichier de structure de la base au nouveau format.
- 4D convertit le fichier de données de la base au nouveau format.

Une fois ces opérations réalisées, pendant l'utilisation de la base, si un formulaire n'a pas été modifié en mode Structure, il est toujours stocké dans le fichier de structure à la manière de la V5. Pour assurer la compatibilité avec vos applications V5, les propriétés d'événements des objets et des formulaires sont définies automatiquement de manière à correspondre aux paramétrages de la V5. Cela signifie que les événements listés ci-dessous (dans la première colonne) seront automatiquement sélectionnés et que les "anciennes" commandes version 5 (dans la deuxième colonne) se comporteront comme dans la version 5 :

Evénements V6	**Cycles d'exécution du formulaire V5**	**Commande V5**
Sur chargement	Phase avant	Avant
Sur validation	Phase après	Apres
Sur clic souris	Phase pendant générique	Pendant
Sur double clic souris	Phase pendant générique	Pendant
Sur activation	Phase activation	Activation
Sur désactivation	Phase désactivation	Desactivation
Sur appel extérieur	Phase appel extérieur	Appel exterieur
Sur menu sélectionné	Phase pendant générique	Pendant plus Menu choisi
Sur données modifiées	Phase pendant générique	Pendant
Sur appel zone du plug in	Phase pendant générique	Pendant
Sur entête	Phase d'impression d'en-tête	En entete
Sur impression corps	Phase pendant générique	Pendant
Sur impression sous total	Phase d'impression des ruptures	En rupture
Sur impression pied de page	Phase d'impression de pied de page	En pied
Sur affichage corps	Phase avant et pendant	Avant & Pendant
Sur ouverture corps	Phase pendant générique	Pendant
Sur fermeture corps	Phase pendant générique	Pendant

Si l'option N'exécuter que pendant était sélectionnée pour un objet (champ ou variable) en version 5, les propriétés d'événements de l'objet sont réduites à celles qui correspondent au cycle d'exécution Pendant qui pouvait survenir pendant la saisie des données en version 5 :

Evénements V6	**Cycles d'exécution du formulaire V5**	**Commande V5**
Sur clic souris	Phase pendant générique	Pendant
Sur double clic souris	Phase pendant générique	Pendant
Sur données modifiées	Phase pendant générique	Pendant
Sur appel zone du plug in	Phase pendant générique	Pendant

Lorsque vous avez modifié un formulaire et ses objets en V6, les propriétés d'événements du formulaire et des objets sont fixées par défaut par rapport aux mêmes principes. Afin de bénéficier des nouveaux événements introduits dans la V6, vous pouvez ensuite sélectionner des propriétés d'événements pour le formulaire et ses objets en mode Structure et modifier les méthodes formulaire et objet à l'aide de la commande Evenement formulaire.

Les nouveaux événements sans cycles d'exécution V5 correspondants sont les suivants :

Evénements V6	**Cycles d'exécution du formulaire V5**	**Commande V5**
Sur libération	Pas de cycle d'exécution équivalent	Aucune
Sur entrée clavier	Pas de cycle d'exécution équivalent	Aucune
Sur gain focus	Pas de cycle d'exécution équivalent	Aucune
Sur perte focus	Pas de cycle d'exécution équivalent	Aucune
Sur déposer	Pas de cycle d'exécution équivalent	Aucune
Sur glisser	Pas de cycle d'exécution équivalent	Aucune
Sur case de fermeture	Pas de cycle d'exécution équivalent	Creer fenetre (avec case de fermeture)

Les nouveaux événements qui vous permettent de mieux exécuter des actions en fonction des événements sont les suivants :

Evénements V6	**Cycles d'exécution du formulaire V5**	**Commande V5**
Sur clic souris	Phase pendant générique	Pendant
Sur double clic souris	Phase pendant générique	Pendant
Sur menu sélectionné	Phase pendant générique	Pendant + Menu choisi
Sur données modifiées	Phase pendant générique	Pendant
Sur appel zone du plug in	Phase pendant générique	Pendant
Sur impression corps	Phase pendant générique	Pendant
Sur affichage corps	Phase avant et pendant	Avant & Pendant
Sur ouverture corps	Phase pendant générique	Pendant
Sur fermeture corps	Phase pendant générique	Pendant

Exemples

Dans tous les exemples ci-dessous, nous supposons que les propriétés d'événements des formulaires et des objets ont été sélectionnées de manière appropriée.

(1) L'exemple suivant trie une sélection de sous-enregistrements pour la sous-table [Parents]Enfants avant qu'un formulaire de la table [Parents] soit affiché à l'écran :

```
      ` Méthode d'un formulaire pour la table [Parents]
   Au cas ou
⇒     : (Evenement formulaire=Sur chargement)
         TRIER SOUS ENREGISTREMENTS([Parents]Enfants;[Parents]Enfants'Prénom;>)
         ` ...
   Fin de cas
```

(2) L'exemple suivant montre l'utilisation de l'événement Sur validation pour affecter automatiquement la date lorsque l'enregistrement est modifié :

```
      ` Méthode d'un formulaire
   Au cas ou
         ` ...
⇒     : (Evenement formulaire=Sur validation)
         [LaTable]Date de modification:=Date du jour
   Fin de cas
```

(3) Dans l'exemple suivant, la gestion complète d'un menu déroulant (initialisation, clics et relâchement de l'objet) est placée dans la méthode de l'objet :

```
      ` Méthode objet du menu déroulant taTaille
   Au cas ou
⇒     : (Evenement formulaire=Sur chargement)
         TABLEAU ALPHA(31;taTaille;3)
         taTaille{1}:="Petit"
         taTaille{1}:="Moyen"
         taTaille{1}:="Grand"
⇒     : (Evenement formulaire=Sur clic souris)
         Si (taTaille#0)
            ALERTE("Vous avez choisi la taille "+taTaille{taTaille}+".")
         Fin de si
⇒     : (Evenement formulaire=Sur libération)
         EFFACER VARIABLE(taTaille)
   Fin de cas
```

(4) L'exemple suivant montre comment, dans une méthode objet, gérer et valider l'opération de glisser-déposer à partir d'un champ qui n'accepte que des images.

```
      ` Méthode objet du champ Image [LaTable]uneImage
   Au cas ou
⇒     : (Evenement formulaire=Sur glisser)
         ` On est en train de glisser-déposer un objet et la souris est au-dessus d'un
         ` champ. Obtenir les informations sur l'objet source
         PROPRIETES GLISSER DEPOSER ($vpObjetSource;$vlElémentSource;
                                                              $lProcessSource)
         ` Notez que nous n'avons pas besoin de tester le numéro de process source
         ` pour la méthode objet exécutée parce qu'elle est dans le même process
         $vlTypeDonnées:=Type ($vpSrcObject->)
         ` Les données source sont-elles une image (champ, variable ou tableau) ?
         Si (($vlTypeDonnées=Est une image) | ($vlTypeDonnées=Est un tableau image))
            ` Accepter l'opération
            ` Notez que le bouton de la souris est toujours enfoncé, 4D affiche l'objet
            ` en inverse vidéo afin d'informer l'utilisateur que les données source
            ` peuvent être déposées sur cet objet
            $0:=0
         Sinon
            ` Sinon, refuser l'opération
            $0:=-1
            ` Dans ce cas, l'objet n'est pas affiché en inverse vidéo
         Fin de si
⇒     : (Evenement formulaire=Sur déposer)
         ` Les données source ont été déposées sur l'objet, donc nous avons besoin de
         ` les copier dans l'objet. Obtenir les informations sur l'objet source
         PROPRIETES GLISSER DEPOSER ($vpObjetSource;$vlElémentSource;
                                                              $lProcessSource)
         $vlTypeDonnées:=Type ($vpSrcObject->)
         Au cas ou
            ` L'objet source est un champ ou une variable de type Image
         : ($vlTypeDonnées=Est une image)
               ` Est-ce que l'objet source est dans le même process (dans la même
               ` fenêtre et le même formulaire) ?
            Si ($lProcessSource=Numero du process courant)
                  ` Copier la valeur source
               [LaTable]uneImage:=$vpObjetSource->
            Sinon
                  ` Sinon, est-ce que l'objet source est une variable ?
               Si (Est une variable ($vpObjetSource))
```

```
                        ` Obtenir la valeur du process source
                    LIRE VARIABLE PROCESS ($lProcessSource;$vpObjetSource->
                                                        ;$vgImageGlissée)
                    [LaTable]uneImage:=$vgImageGlissée
                Sinon
                    ` Sinon, utiliser APPELER PROCESS pour obtenir la valeur du champ
                    ` du process source
                Fin de si
            Fin de si
                ` L'objet source est un tableau de type Image
        : ($vlTypeDonnées=Est un tableau image)
                ` Est-ce que l'objet source est dans le même process (dans la même
                ` fenêtre et le même formulaire) ?
            Si ($lProcessSource=Numero du process courant)
                    ` Copier la valeur source
                [LaTable]uneImage:=$vpSrcObject->{$vlElémentSource}
            Sinon
                    ` Sinon, obtenir la valeur du process source
                LIRE VARIABLE PROCESS ($lProcessSource;$vpObjetSource
                        ->{$vlElémentSource};$vgImageGlissée)
                [LaTable]uneImage:=$vgImageGlissée
            Fin de si
        Fin de cas
    Fin de cas
```

Note : Pour d'autres exemples sur la gestion des événements Sur glisser et Sur déposer, référez-vous aux exemples de la commande PROPRIETES GLISSER DEPOSER.

(5) L'exemple suivant est une méthode formulaire générique. Elle fait apparaître chacun des événements qui peuvent survenir lorsqu'un formulaire est utilisé comme formulaire sortie :

```
    ` Méthode formulaire d'un formulaire sortie
$vpFormTable:=Table du formulaire courant
Au cas ou
        ` ...
⇒       : (Evenement formulaire=Sur entête)
            ` La zone en-tête va être imprimée ou affichée
        Au cas ou
            : (Avant selection($vpFormTable->))
                ` Le code pour la première rupture d'en-tête doit être placé ici
            : (Niveau = 1)
                ` Le code pour la rupture d'en-tête niveau 1 doit être placé ici
```

```
            : (Niveau = 2)
               ` Le code pour la rupture d'en-tête niveau 2 doit être placé ici
               ` ...
         Fin de cas
⇒        : (Evenement formulaire=Sur impression corps)
            ` Un enregistrement va être imprimé
            ` Le code pour chaque enregistrement doit être placé ici
⇒        : (Evenement formulaire=Sur impression sous total)
            ` Une rupture va être imprimée
         Au cas ou
            : (Niveau = 0)
               ` Le code pour la rupture 0 doit être placé ici
            : (Niveau = 1)
               ` Le code pour la rupture 1 doit être placé ici
               ` ...
         Fin de cas
⇒        : (Evenement formulaire=Sur impression pied de page)
         Si (Fin de selection($vpFormTable->))
            ` Le code pour le dernier pied de page doit être placé ici
         Sinon
            ` Le code pour le pied de page doit être placé ici
         Fin de si
   Fin de cas
```

(6) L'exemple suivant montre une méthode formulaire générique qui gère les événements pouvant survenir dans un formulaire sortie quand il s'affiche à l'aide de VISUALISER SELECTION ou MODIFIER SELECTION. Dans un but informatif, elle affiche l'événement dans la barre de titre de la fenêtre.

```
   ` Une méthode formulaire exemple
   Au cas ou
⇒     : (Evenement formulaire=Sur chargement)
         $vaEvénement:="Le formulaire va être affiché"
⇒     : (Evenement formulaire=Sur libération)
         $vaEvénement:="Le formulaire sortie est refermé et va disparaître de l'écran"
⇒     : (Evenement formulaire=Sur affichage corps)
         $vaEvénement:="Affichage de l'enregistrement n°"
                                          +Chaine(Numero dans selection([LaTable]))
⇒     : (Evenement formulaire=Sur menu sélectionné)
         $vaEvénement:="Une commande de menu a été sélectionnée"
⇒     : (Evenement formulaire=Sur entête")
         $vaEvénement:="L'en-tête va être imprimé ou affiché"
```

```
⇒       : (Evenement formulaire=Sur clic souris)
            $vaEvénement:="On a cliqué sur un enregistrement"
⇒       : (Evenement formulaire=Sur double clic souris)
            $vaEvénement:="On a double-cliqué sur un enregistrement"
⇒       : (Evenement formulaire=Sur ouverture corps)
            $vaEvénement:="On a double-cliqué sur l'enregistrement n°"+Chaine
                                                (Numero dans selection([LaTable]))
⇒       : (Evenement formulaire=Sur fermeture corps)
            $vaEvénement:="Retour au formulaire sortie"
⇒       : (Evenement formulaire=Sur activation)
            $vaEvénement:="La fenêtre du formulaire passe au premier plan"
⇒       : (Evenement formulaire=Sur désactivation)
            $vaEvénement:="La fenêtre du formulaire n'est plus au premier plan"
⇒       : (Evenement formulaire=Sur menu sélectionné)
            $vaEvénement:="Une ligne de menu a été sélectionnée"
⇒       : (Evenement formulaire=Sur appel extérieur)
            $vaEvénement:="Un appel extérieur a été reçu"
        Sinon
⇒           $vaEvénement:="Que se passe-t-il ? L'événement n°"
                                                +Chaine(Evenement formulaire)
    Fin de cas
    CHANGER TITRE FENETRE ($vaEvénement)
```

(7) Pour des exemples de gestion des événements Sur avant frappe clavier et Sur après frappe clavier, référez-vous aux exemples des commandes Lire texte edite, Frappe clavier et FILTRER FRAPPE CLAVIER.

(8) L'exemple suivant montre comment traiter de la même manière les clics et double-clics dans une zone de défilement :

```
    ` Méthode objet pour la zone de défilement taChoix
    Au cas ou
⇒       : (Evenement formulaire=Sur chargement)
            TABLEAU ALPHA (...;taChoix;...)
                ` ...
            taChoix:=0
⇒       : ((Evenement formulaire=Sur clic souris) | (Evenement formulaire
                                                =Sur double clic souris))
            Si (taChoix#0)
                    ` On a cliqué sur un élément, faire quelque chose...
            Fin de si
                ` ...
    Fin de cas
```

(9) L'exemple suivant montre comment traiter les clics et double-clics de manière différente (notez l'utilisation de l'élément zéro pour conserver la valeur de l'élément sélectionné) :

```
      ` Méthode objet pour la zone de défilement tonChoix
   Au cas ou
⇒     : (Evenement formulaire=Sur chargement)
         TABLEAU ALPHA (...;taChoix;...)
            ` ...
         taChoix:=0
         taChoix{0}:="0"
⇒     : (Evenement formulaire=Sur clic souris)
         Si (taChoix#0)
            Si (taChoix#Num(taChoix))
                  ` On a cliqué sur un élément, faire quelque chose
               ` ...
               ` Sauvegarder l'élément nouvellement sélectionné pour la prochaine fois
               taChoix{0}:=Chaine(taChoix)
            Fin de si
         Sinon
            taChoix:=Num(taChoix{0})
         Fin de si
⇒     : (Evenement formulaire=Sur double clic souris)
         Si (taChoix#0)
               ` On a double-cliqué sur un élément, faire quelque chose
         Fin de si
            ` ...
   Fin de cas
```

(10) L'exemple suivant montre comment maintenir une zone contenant du texte à partir d'une méthode formulaire à l'aide des événements Sur gain focus et Sur perte focus :

```
      ` Méthode formulaire [Contacts];"Entrée"
   Au cas ou
⇒     : (Evenement formulaire=Sur chargement)
         C_TEXTE(vtZoneEtat)
         vtZoneEtat:=""
⇒     : (Evenement formulaire=Sur gain focus)
         RESOUDRE POINTEUR(Dernier objet;$vsNomVar;$vlNumTable;$vlNumChamp)
         Si (($vlNumTable#0) & ($vlNumChamp#0))
            Au cas ou
               : ($vlNumChamp=1) ` Champ nom
                  vtZoneEtat:="Saisissez le nom du contact, il sera automatiquement mis
                                                                        en majuscules."
                     ` ...
               : ($vlNumChamp=10) ` Champ code postal
                  vtZoneEtat:="Saisissez un code postal, il sera automatiquement vérifié et
                                                                        validé."
                     ` ...
            Fin de cas
         Fin de si
⇒     : (Evenement formulaire=Sur perte focus)
         vtZoneEtat:=""
            ` ...
   Fin de cas
```

(11) L'exemple suivant montre comment traiter l'événement de fermeture de fenêtre avec un formulaire utilisé pour l'entrée des données :

```
   ` Méthode pour un formulaire entrée
$vpFormulaireTable:=Table du formulaire courant
Au cas ou
      ` ...
⇒     : (Evenement formulaire=Sur case de fermeture)
         Si (Enregistrement modifie($vpFormulaireTable->))
            CONFIRMER ("Cet enregistrement a été modifié. Voulez-vous sauvegarder les
                                                                  modifications ?")
            Si (OK=1)
               VALIDER
            Sinon
               NE PAS VALIDER
            Fin de si
         Sinon
            NE PAS VALIDER
         Fin de si
            ` ...
Fin de cas
```

(12) L'exemple suivant montre comment mettre en majuscules un champ Texte ou Alphanumérique chaque fois que la valeur est modifiée :

```
   ` Méthode objet pour [Contacts]Prénom
Au cas ou
      ` ...
   : (Evenement formulaire=Sur données modifiées)
      [Contacts]Prénom:= Majusc(Sous chaine([Contacts]Prénom;1;1))
                                    +Minusc(Sous chaine([Contacts]Prénom;2))
         ` ...
Fin de cas
```

Référence

APPELER PROCESS, FILTRER FRAPPE CLAVIER, FIXER MINUTEUR, Frappe clavier, Lire texte edite, PROPRIETES GLISSER DEPOSER, Table du formulaire courant.

version 5

Note de compatibilité
Cette fonction n'est conservée que pour des raisons de compatibilité avec les versions précédentes de 4D. A compter de la version 6, il est préférable d'utiliser la fonction Evenement formulaire et de tester si elle retourne l'événement Sur chargement.

Avant → Booléen

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

Si vous souhaitez que la phase Avant du cycle d'exécution soit générée, assurez-vous que l'événement Sur chargement a bien été sélectionné, en mode Structure, dans les propriétés du formulaire et/ou des objets concernés.

Référence

Evenement formulaire.

Pendant

Evénements formulaire

version 5

Note de compatibilité
Cette fonction n'est conservée que pour des raisons de compatibilité avec les versions précédentes de 4D. A compter de la version 6, il est préférable d'utiliser la fonction Evenement formulaire et de tester si elle retourne un événement tel que Sur clic souris.

Pendant → Booléen

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

Si vous souhaitez que la phase Pendant du cycle d'exécution soit générée, assurez-vous qu'au moins un événement tel que Sur clic souris a bien été sélectionné, en mode Structure, dans les propriétés du formulaire et/ou des objets concernés.

Référence

Evenement formulaire.

version 5

Note de compatibilité
Cette fonction n'est conservée que pour des raisons de compatibilité avec les versions précédentes de 4D. A compter de la version 6, il est préférable d'utiliser la fonction Evenement formulaire et de tester si elle retourne un événement tel que Sur validation.

Apres → Booléen

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

Si vous souhaitez que la phase Apres du cycle d'exécution soit générée, assurez-vous que l'événement Sur validation a bien été sélectionné, en mode Structure, dans les propriétés du formulaire et/ou des objets concernés.

Référence

Evenement formulaire.

version 5

Note de compatibilité
Cette commande a été conservée pour des raisons de compatibilité. A partir de la version 6 de 4D, il est préférable d'utiliser la commande Evenement formulaire et de tester si elle retourne l'événement Sur entête.

En entete → Booléen

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

Si vous souhaitez que le cycle d'exécution En entete soit généré, assurez-vous que l'événement formulaire Sur entête a bien été activé dans la boîte de dialogue des propriétés du formulaire ou du (des) objet(s) sélectionné(s), en mode Structure.

Référence

En pied, En rupture, Pendant.

version 5

Note de compatibilité
Cette commande a été conservée pour des raisons de compatibilité. A partir de la version 6 de 4D, il est préférable d'utiliser la commande Evenement formulaire et de tester si elle retourne l'événement Sur impression sous total.

En rupture → Booléen

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description
Si vous souhaitez que le cycle d'exécution En rupture soit généré, assurez-vous que l'événement formulaire Sur impression sous total a bien été activé dans la boîte de dialogue des propriétés du formulaire ou du (des) objet(s) sélectionné(s), en mode Structure.

Référence
En entete, En pied, Pendant.

version 5

Note de compatibilité
Cette fonction n'est conservée que pour des raisons de compatibilité avec les versions précédentes de 4D. A compter de la version 6, il est préférable d'utiliser la fonction Evenement formulaire et de tester si elle retourne l'événement Sur impression pied de page.

En pied → Booléen

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description
Si vous voulez que le cycle d'exécution En pied soit généré, vérifiez que la propriété d'événement Sur impression pied de page du formulaire et/ou des objets est sélectionnée en mode Structure.

Référence
En entete, En rupture, Pendant.

Activation

Evénements formulaire

version 5

Note de compatibilité
Cette fonction n'est conservée que pour des raisons de compatibilité avec les versions précédentes de 4D. A compter de la version 6, il est préférable d'utiliser la fonction Evenement formulaire et de tester si elle retourne l'événement Sur activation.

Activation → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Vrai si le cycle d'exécution est en activation

Description

Activation retourne Vrai dans une méthode formulaire lorsque la fenêtre contenant le formulaire passe au premier plan.

ATTENTION : N'appelez pas de commandes telles que TRACE ou ALERTE dans la phase Activation d'un formulaire, car cela provoquerait une boucle sans fin.

Note : Si vous voulez que le cycle d'exécution Activation soit généré, assurez-vous que la propriété d'événement Sur activation du formulaire et/ou des objet(s) est sélectionnée en mode Structure. Cette propriété est automatiquement définie pour les bases converties.

Référence

Desactivation, Evenement formulaire.

Note de compatibilité
Cette fonction n'est conservée que pour des raisons de compatibilité avec les versions précédentes de 4D. A compter de la version 6, il est préférable d'utiliser la fonction Evenement formulaire et de tester si elle retourne l'événement Sur désactivation.

Desactivation → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Vrai si le cycle d'exécution est en désactivation

Description

Desactivation retourne Vrai dans une méthode formulaire ou méthode objet lorsque la fenêtre appartenant au process du premier plan, contenant le formulaire, passe à l'arrière-plan.

Si vous voulez que le cycle d'exécution Desactivation soit généré, vérifiez que la propriété d'événement Sur désactivation du formulaire et/ou de l'objet est sélectionnée en mode Structure.

Référence

Activation, Evenement formulaire.

Appel exterieur

Evénements formulaire

version 5

Note de compatibilité
Cette fonction n'est conservée que pour des raisons de compatibilité avec les versions précédentes de 4D. A compter de la version 6, il est préférable d'utiliser la fonction Evenement formulaire et de tester si elle retourne un événement tel que Sur appel exterieur.

Appel exterieur → Booléen

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description
Si vous voulez que le cycle d'exécution Appel exterieur soit généré, vérifiez que la propriété d'événement Sur appel extérieur du formulaire et/ou des objets est sélectionnée en mode Structure.

Référence
APPELER PROCESS, Evenement formulaire.

Lire texte edite → Texte

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Texte	←	Texte en cours de saisie

Description

La commande Lire texte edite retourne le texte en cours de saisie dans un objet de formulaire.
Cette commande est principalement destinée à être utilisée avec le nouvel événement formulaire Sur après frappe clavier pour récupérer le texte au fur et à mesure de la frappe. Elle peut également être utilisée avec l'événement formulaire Sur avant frappe clavier.

Note : Pour des raisons d'harmonisation avec le nouvel événement formulaire Sur après frappe clavier, l'événement Sur entrée clavier a été renommé en Sur avant frappe clavier à compter de la version 6.5 de 4D.

La combinaison de cette commande avec les événements formulaire Sur avant frappe clavier et Sur après frappe clavier fonctionne de la manière suivante :
• Dès qu'un caractère est tapé au clavier, l'événement Sur avant frappe clavier est généré. Dans cet événement, la fonction Lire texte edite retourne le contenu de la zone avant la dernière frappe clavier. Par exemple, si la zone contient "PA" et que l'utilisateur tape "R", Lire texte edite retourne "PA" dans l'événement Sur avant frappe clavier. Si la zone ne contient rien au départ, Lire texte edite retourne une chaîne vide.
• Ensuite, l'événement formulaire Sur après frappe clavier est généré. Dans cet événement, la fonction Lire texte edite retourne le contenu de la zone y compris le dernier caractère entré au clavier. Par exemple, si la zone contient "PA" et que l'utilisateur tape "R", Lire texte edite retourne "PAR" dans l'événement Sur après frappe clavier.

Ces deux événements ne sont générés que dans les méthodes des objets concernés. Dans un contexte autre que la saisie dans un formulaire, cette fonction retourne une chaîne vide.

Exemples

(1) Dans un formulaire entrée, vous souhaitez que les caractères saisis soient automatiquement mis en majuscules :

```
     Si (Evenement formulaire=Sur apres frappe clavier)
⇒       [Voyages]Agences:=Majusc(Lire texte edite)
     Fin de si
```

(2) Voici un exemple de traitement à la volée des caractères saisis dans un champ texte. Le principe consiste à placer dans un autre champ texte (appelé “Mots”) la décomposition en mots de la phrase en cours de saisie. Pour cela, écrivez dans la méthode objet du champ de saisie :

```
    Si (Evenement formulaire=Sur apres frappe clavier)
⇒       $SaisieTempsRéel:=Lire texte edite
        PROPRIETES PLATE FORME ($plate_forme)
        Si ($plate_forme#3)    ` Macintosh ou Power Macintosh
            Repeter
                $PhraseDécomposée:=Remplacer chaine($SaisieTempsRéel;Caractere(32);
                                                                    Caractere(13))
            Jusque (Position(" ";$PhraseDécomposée)=0)
        Sinon    ` Windows
            Repeter
            $PhraseDécomposée:=Remplacer chaine($SaisieTempsRéel;Caractere(32);
                                                    Caractere(13)+Caractere(10))
            Jusque (Position(" ";$PhraseDécomposée)=0)
        Fin de si
        [Exemple]Mots:=$PhraseDécomposée
    Fin de si
```

Note : Cet exemple n’est pas exhaustif puisque l’on considère que les mots sont séparés par des espaces uniquement (Caractere (32)). La mise au point d’un système complet nécessiterait l’ajout d’autres filtres afin de repérer tous les mots (point-virgules, virgules, apostrophes, etc...).

Référence

Evenement formulaire.

FIXER MINUTEUR (tickCount)

Paramètre	Type		Description
tickCount	Entier long	→	Nombre de ticks

Description

La commande FIXER MINUTEUR permet d'activer l'événement formulaire Sur minuteur et de fixer, pour le process courant, le nombre de ticks (1 tick = 1/60ème de seconde) entre chaque événement formulaire Sur minuteur.

Note : Pour plus d'informations sur cet événement formulaire, reportez-vous à la description de la commande Evenement formulaire.

Si elle est appelée dans un contexte autre que l'affichage d'un formulaire, cette commande ne fait rien.

Note serveur Web : Le serveur Web 4D peut tirer parti de cette commande ainsi que de l'événement formulaire Sur minuteur pour réafficher des formulaires 4D en mode contextuel. Ce fonctionnement permet d'obtenir l'envoi de pages HTML mises à jour "en temps réel", tout en économisant la bande passante. En effet, dans ce cas la mise à jour du formulaire n'est pas automatique, il vous faut pour cela appeler la commande REDESSINER. Il est donc possible d'optimiser le système en n'appelant REDESSINER que lorsque les données ont été modifiées.
Seuls les browsers qui interprètent le JavaScript permettront le redessinement automatique. La période définie par FIXER MINUTEUR sera utilisée par le browser ; elle doit être comprise entre quelques secondes (5 étant une valeur pratique) et le timeout du process Web. Reportez-vous à l'exemple n°2.

Pour inactiver par programmation le déclenchement de l'événement formulaire Sur minuteur, appelez de nouveau la commande FIXER MINUTEUR en passant 0 dans le paramètre nbTicks.

Exemples

(1) Vous souhaitez que, lorsqu'un formulaire est affiché à l'écran, un bip soit émis toutes les trois secondes. Pour cela, écrivez dans la méthode du formulaire :

```
    Si (Evenement formulaire=Sur chargement)
⇒       FIXER MINUTEUR(60*3)
    Fin de si
    ...
    Si (Evenement formulaire=Sur minuteur)
       BEEP
    Fin de si
```

(2) Vous souhaitez que votre serveur Web provoque la mise à jour d'un formulaire 4D sur les browsers toutes les cinq secondes. Vous pouvez écrire, dans la méthode du formulaire :

```
    Si (Evenement formulaire=Sur chargement)
⇒       FIXER MINUTEUR(60*5)
    Fin de si
    ...
    Si (Evenement formulaire=Sur minuteur)
       ... `Vous pouvez placer ici un test sur la modification des données et
           `n'exécuter la ligne suivante que si les données ont été modifiées
       REDESSINER ([MaTable];"MonFormulaire")
    Fin de si
```

Référence

Evenement formulaire, REDESSINER.

19

Fenêtres

Le rôle des fenêtres est d'afficher des informations pour l'utilisateur. Trois principales actions nécessitent l'affichage d'une fenêtre : la saisie de données, l'affichage de données et l'affichage de messages destinés à l'utilisateur.

Il y a toujours au moins une fenêtre ouverte à l'écran. Si nécessaire, des barres de défilement sont ajoutées, afin de permettre à l'utilisateur de faire défiler le contenu d'un formulaire lorsque celui-ci est plus grand que la fenêtre. En mode Utilisation, cette fenêtre sera soit la fenêtre vous présentant vos enregistrements sous forme de liste (formulaire sortie), soit la fenêtre vous proposant un enregistrement en saisie (formulaire sortie). En mode Menus créés, cette fenêtre sera l'écran présentant par défaut le logo de 4e Dimension.

Lorsque vous sélectionnez une commande de menu en mode Menus créés, la fenêtre d'accueil peut être effacée et remplacée par des données lorsque vous faites appel aux commandes qui affichent des formulaires. Une fois que l'exécution de ces commandes est terminée, l'écran d'accueil apparaît de nouveau.

Pour créer vos propres fenêtres, faites appel à la commande Creer fenetre. Il existe différents types de fenêtres personnalisées. Lorsque vous n'en avez plus besoin, vous pouvez fermer une fenêtre personnalisée avec la commande FERMER FENETRE, ou en double-cliquant sur la case du menu Système (Windows) ou sur la case de fermeture (MacOS), s'il y en a une.

Certaines commandes, telles que GRAPHE SUR SELECTION, ETAT ou encore IMPRIMER ETIQUETTES ouvrent leurs propres fenêtres et les placent au premier plan.

Si vous démarrez un nouveau process et n'ouvrez pas de fenêtre au démarrage de la méthode process, 4D en créera une automatiquement avec le type par défaut, dès qu'il sera nécessaire d'afficher un formulaire.

Référence

Creer fenetre, Types de fenêtres.

Vous spécifiez le type de fenêtre à ouvrir avec Creer fenetre à l'aide d'une des constantes prédéfinies suivantes :

Constante	Type	Valeur	Fenêtre flottante
Fenêtre standard	Entier long	8	Non
Fenêtre standard sans zoom	Entier long	0	Non
Fenêtre standard de taille fixe	Entier long	4	Non
Dialogue modal	Entier long	1	Non
Dialogue modal déplaçable	Entier long	5	Oui
Dialogue simple	Entier long	2	Oui
Dialogue ombré	Entier long	3	Oui
Fenêtre palette	Entier long	1984	Oui
Fenêtre à coins arrondis	Entier long	16	Non

Fenêtres flottantes

Si vous passez une de ces constantes à Creer fenetre, vous créez une fenêtre standard. Pour ouvrir une fenêtre flottante, passez un type de fenêtre négatif à Creer fenetre.

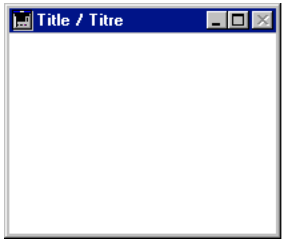
Fenêtres modales

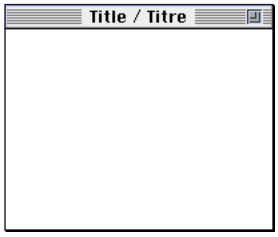
Une fenêtre modale place l'utilisateur dans un état (ou “mode”) où il ne peut agir qu'à l'intérieur de la fenêtre. Tant qu'une fenêtre modale est affichée, les commandes de menu et les autres fenêtres de l'application sont inaccessibles. Pour refermer une fenêtre modale, l'utilisateur doit la valider, l'annuler ou utiliser une des options qu'elle propose. Les boîtes de dialogue d'alerte sont des exemples typiques de fenêtres modales.
Dans 4D, les fenêtres de type 1 et 5 sont modales.

Note : Une fenêtre modale reste toujours au premier plan. Par conséquent, lorsqu'une fenêtre modale appelle une fenêtre non modale, cette dernière s'affiche en arrière-plan, bien qu'elle ait été appelée postérieurement. Il ne faut donc pas effectuer ce type d'opération.
En revanche, lorsqu'une fenêtre modale appelle une autre fenêtre modale, cette dernière s'affiche au premier plan

Vous trouverez ci-dessous une description de chaque type de fenêtre, sous Windows (à gauche) et MacOS (à droite).

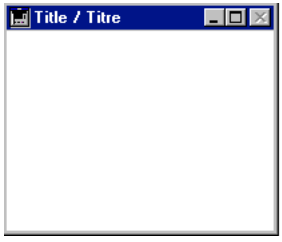
Fenêtre standard (8)

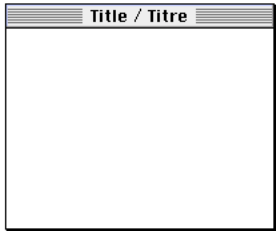




• Peut avoir un titre : Oui
• Peut comporter une case de fermeture ou un équivalent : Oui
• Peut être redimensionnée : Oui
• Peut être agrandie/réduite ou "zoomée" : Oui
• Adaptée aux barres de défilement : Oui
• Utilisation : saisie de données avec des barres de défilement, VISUALISER SELECTION, MODIFIER SELECTION, etc.

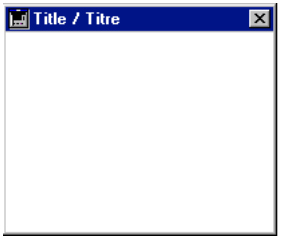
Fenêtre standard sans zoom (0)

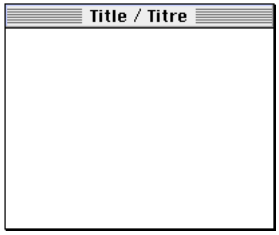




• Peut avoir un titre : Oui
• Peut comporter une case de fermeture ou un équivalent : Oui
• Peut être redimensionnée : Oui
• Peut être agrandie/réduite ou "zoomée" : Non sous MacOS
• Adaptée aux barres de défilement : Oui
• Utilisation : saisie de données avec des barres de défilement, VISUALISER SELECTION, MODIFIER SELECTION, etc.

Fenêtre standard de taille fixe (4)



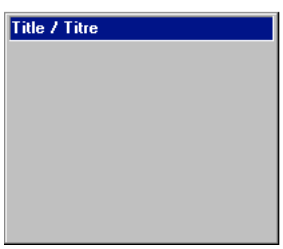


- Peut avoir un titre : Oui
- Peut comporter une case de fermeture ou un équivalent : Oui
- Peut être redimensionnée : Non sous MacOS
- Peut être agrandie/réduite ou "zoomée" : Non
- Adaptée aux barres de défilement : Oui et Non
- Utilisation : saisie de données par AJOUTER ENREGISTREMENT(...;...*) ou équivalent

Dialogue modal (1)

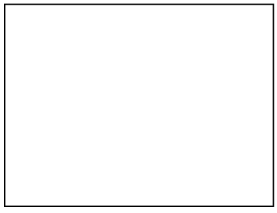
- Peut avoir un titre : Non
- Peut comporter une case de fermeture ou un équivalent : Non
- Peut être redimensionnée : Non
- Peut être agrandie/réduite ou "zoomée" : Non
- Adaptée aux barres de défilement : Non
- Utilisation : DIALOGUE, AJOUTER ENREGISTREMENT(...;...*) ou équivalent
- Les fenêtres de ce type sont modales

Dialogue modal déplaçable (5)



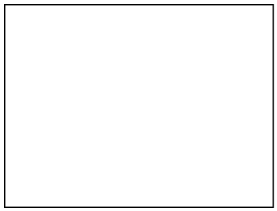
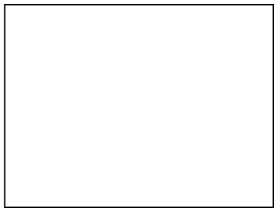
• Peut avoir un titre : Oui
• Peut comporter une case de fermeture ou un équivalent : Non
• Peut être redimensionnée : Non
• Peut être agrandie/réduite ou "zoomée" : Non
• Adaptée aux barres de défilement : Non
• Utilisation : DIALOGUE, AJOUTER ENREGISTREMENT(...;...*) ou équivalent
• Les fenêtres de ce type sont modales mais peuvent être déplacées et utilisées comme fenêtres flottantes

Dialogue ombré (3)



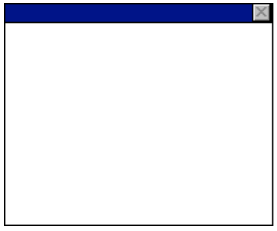
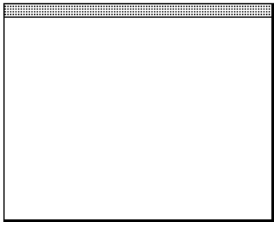
• Peut avoir un titre : Non
• Peut comporter une case de fermeture ou un équivalent : Non
• Peut être redimensionnée : Non
• Peut être agrandie/réduite ou "zoomée" : Non
• Adaptée aux barres de défilement : Non
• Utilisation : DIALOGUE, AJOUTER ENREGISTREMENT(...;...*) ou équivalent, sous MacOS (non standard sous Windows).

Dialogue simple (2)

• Peut avoir un titre : Non
• Peut comporter une case de fermeture ou un équivalent : Non
• Peut être redimensionnée : Non
• Peut être agrandie/réduite ou "zoomée" : Non
• Adaptée aux barres de défilement : Non
• Utilisation : DIALOGUE, AJOUTER ENREGISTREMENT(...;...*) ou équivalent, sous MacOS (non standard sous Windows).

Fenêtre palette (1984 {+ 1} {+ 2} {+ 4} {+ 8})

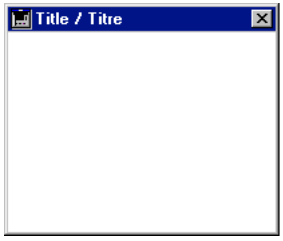
Lorsque vous appelez Creer fenetre, vous pouvez additionner une ou plusieurs constantes supplémentaires (listées ci-dessous) à la constante Fenêtre palette afin de créer des variantes ayant des comportements différents :

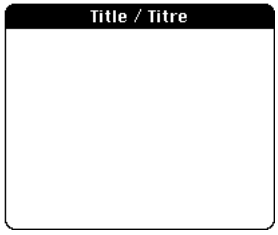
Constante	Type	Valeur
Avec case de zoom	Entier long	8
Avec case de contrôle de taille	Entier long	4
Avec titre de fenêtre	Entier long	2
Avec barre de titre active	Entier long	1

• Peut avoir un titre : Oui si la variante Avec titre de fenêtre est spécifiée
• Peut comporter une case de fermeture ou un équivalent : Oui
• Peut être redimensionnée : Oui si la variante Avec case de contrôle de taille est spécifiée
• Peut être agrandie/réduite ou "zoomée" : Oui si la variante Avec case de zoom est spécifiée

• Adaptée aux barres de défilement : Oui si la variante Avec case de contrôle de taille est spécifiée
• Utilisation : fenêtres flottantes avec DIALOGUE ou VISUALISER SELECTION (pas de saisie de données)

Fenêtre à coins arrondis (16)





• Peut avoir un titre : Oui
• Peut comporter une case de fermeture ou un équivalent : Oui
• Peut être redimensionnée : Non sous MacOS
• Peut être agrandie/réduite ou "zoomée" : Non
• Adaptée aux barres de défilement : Non
• Utilisation : rare

Référence

Creer fenetre, Creer fenetre externe.

Creer fenetre (gauche; haut; droite; bas{; type{; titre{; caseFermeture}}}){ → RefFen }

Paramètre	Type		Description
gauche	Numérique	→	Coordonnée gauche de l'intérieur de la fenêtre
haut	Numérique	→	Coordonnée supérieure de l'intérieur de la fenêtre
droite	Numérique	→	Coordonnée droite de l'intérieur de la fenêtre
bas	Numérique	→	Coordonnée inférieure de l'intérieur de la fenêtre
type	Numérique	→	Type de la fenêtre
titre	Alpha	→	Titre de la fenêtre
caseFermeture	Alpha	→	Méthode à appeler en cas de double-clic sur la case du menu Système ou de clic sur la case de fermeture
Résultat	RefFen	←	Numéro de référence de la fenêtre

Description

Creer fenetre ouvre une nouvelle fenêtre dont les dimensions sont définies par les quatre premiers paramètres :

• gauche est la distance en pixels entre le côté gauche de la fenêtre de l'application et le côté intérieur gauche de la fenêtre.
• haut est la distance en pixels entre le haut de la fenêtre de l'application et le bord supérieur de l'intérieur de la fenêtre.
• droite est la distance en pixels entre le côté gauche de la fenêtre de l'application et le côté intérieur droit de la fenêtre.
• bas est la distance en pixels entre le haut de la fenêtre de l'application et le bord inférieur de l'intérieur de la fenêtre.

Si vous passez -1 dans droite et bas,vous indiquez à 4D qu'il faut redimensionner automatiquement la fenêtre si les conditions suivantes sont réunies :

• Vous avez conçu un formulaire et défini ses options de dimensionnement dans la fenêtre des propriétés des formulaires du mode Structure.
• Avant d'appeler Creer fenetre vous avez sélectionné le formulaire à l'aide de la commande FORMULAIRE ENTREE, à laquelle vous avez passé le paramètre optionnel *.

Important : Ce dimensionnement automatique de la fenêtre n'aura lieu que si vous avez au préalable appelé la commande FORMULAIRE ENTREE pour le formulaire que vous allez afficher dans la fenêtre, et si vous lui avez passé le paramètre optionnel *.

• Le paramètre type est optionnel. Il définit le type de fenêtre que vous souhaitez afficher, et correspond aux différentes fenêtres présentées dans la section Types de fenêtres. Si le type passé est négatif, la fenêtre sera flottante. Si le type n'est pas spécifié, le type 1 est utilisé par défaut.

• Le paramètre titre indique le titre (optionnel) de la fenêtre.

Si vous passez une chaîne de caractères vide ("") dans titre, vous indiquez à 4D d'utiliser les valeurs saisies dans la zone **Nom de la fenêtre** de la fenêtre des Propriétés du formulaire en mode Structure pour le titre du formulaire que vous allez afficher dans la fenêtre.

Important : Le nom par défaut du formulaire ne sera appliqué à la fenêtre que si vous avez appelé la commande FORMULAIRE ENTREE pour le formulaire que vous allez afficher dans la fenêtre et si vous lui avez passé le paramètre optionnel *.

• Le paramètre caseFermeture, optionnel, désigne la méthode de gestion de la fermeture de la fenêtre. Si ce paramètre est passé, la case du menu Système (sous Windows) ou une case de fermeture (sous MacOS) est ajoutée à la fenêtre. Lorsque l'utilisateur Windows double-clique sur la case du menu Système ou que l'utilisateur MacOS clique sur la case de fermeture, la méthode passée dans caseFermeture est exécutée.

Note version 6 : Vous pouvez aussi gérer la fermeture à partir de la méthode du formulaire affiché dans la fenêtre pendant l'événement Sur case de fermeture. Pour plus d'informations sur ce point, reportez-vous à la commande Evenement formulaire.

Si plusieurs fenêtres sont ouvertes dans le même process, la dernière fenêtre créée est la fenêtre active (de premier plan) du process. Seules les informations situées dans la fenêtre active peuvent être modifiées. Toutes les autres fenêtres peuvent être visualisées. Lorsque l'utilisateur tape une touche du clavier, la fenêtre active vient toujours se placer au premier plan, si elle n'y est pas déjà.

Les formulaires sont affichés à l'intérieur de fenêtres ouvertes à l'écran. Le texte passé à la commande MESSAGE est également affiché dans une fenêtre.

Exemples

(1) La méthode projet suivante ouvre une fenêtre centrée dans la fenêtre principale (sous Windows) ou dans l'écran principal (sous MacOS). Notez qu'elle accepte deux, trois ou quatre paramètres :

```
` Méthode projet OUVRIR FENETRE CENTREE
` $1 – Largeur de la fenêtre
` $2 – Hauteur de la fenêtre
` $3 – Type de la fenêtre (optionnel)
` $4 – Titre de la fenêtre (optionnel)
$SW:=Largeur ecran\2
$SH:=(Hauteur ecran\2)-10
$WW:=$1\2
$WH:=$2\2
Au cas ou
   : (Nombre de parametres=2)
⇒     Creer fenetre($SW-$WW;$SH-$WH;$SW+$WW;$SH+$WH)
   : (Nombre de parametres=3)
⇒     Creer fenetre($SW-$WW;$SH-$WH;$SW+$WW;$SH+$WH;$3)
   : (Nombre de parametres=4)
⇒     Creer fenetre($SW-$WW;$SH-$WH;$SW+$WW;$SH+$WH;$3;$4)
Fin de cas
```

Une fois que cette méthode projet est écrite, vous pouvez l'utiliser de la manière suivante :

```
OUVRIR FENETRE CENTREE (400;250;Dialogue simple;"Mise à jour Archives")
DIALOGUE([Table outils];"OPTIONS MAJ")
FERMER FENETRE
Si (OK=1)
   ` ...
Fin de si
```

(2) L'exemple suivant crée une fenêtre flottante comportant une case de menu système (sous Windows) ou une case de fermeture (sous MacOS). La fenêtre est créée dans le coin supérieur droit de la fenêtre de l'application.

```
⇒  Creer fenetre(Largeur ecran-149;33;Largeur ecran-4;178;-Fenêtre palette;"";
                                                          "caseFermeture")
   DIALOGUE([Dialogues];"Palette de couleurs")
```

La méthode caseFermeture appelle la commande NE PAS VALIDER :

```
NE PAS VALIDER
```

(3) L'exemple suivant ouvre une fenêtre dont le titre et la taille proviennent des propriétés du formulaire affiché dans la fenêtre :

```
      FORMULAIRE ENTREE([Clients];"Ajout d'enregistrements";*)
⇒     Creer fenetre(10;80;-1;-1;Fenêtre standard;"")
      Repeter
         AJOUTER ENREGISTREMENT([Clients])
      Jusque (OK=0)
```

Rappel : Pour que la fonction Creer fenetre utilise automatiquement les propriétés du formulaire, vous devez avoir appelé FORMULAIRE ENTREE avec le paramètre optionnel * et les propriétés du formulaire doivent avoir été définies en fonction de cette utilisation.

Référence

Creer fenetre externe, Creer fenetre formulaire, FERMER FENETRE.

Creer fenetre externe

Fenêtres

version 6.0 (Modifiée)

Creer fenetre externe (gauche; haut; droit; bas; type; titre; zonePlugin) → Numérique

Paramètre	Type		Description
gauche	Numérique	→	Coordonnée gauche de l'intérieur de la fenêtre
haut	Numérique	→	Coordonnée supérieure de l'intérieur de la fenêtre
droit	Numérique	→	Coordonnée droite de l'intérieur de la fenêtre
bas	Numérique	→	Coordonnée inférieure de l'intérieur de la fenêtre
type	Numérique	→	Type de la fenêtre
titre	Alpha	→	Titre de la fenêtre
zonePlugin	Alpha	→	Commande de zone externe
Résultat	Numérique	←	Numéro de référence de la fenêtre

Description

Creer fenetre externe crée une nouvelle fenêtre et affiche la zone externe gérée par la commande zonePlugin fournie par un plug-in 4D.

Creer fenetre externe retourne un Entier long qui peut être utilisé à la fois comme numéro de référence de fenêtre (qui peut être exploité par les autres commandes du thème Fenêtres) et comme référence de la zone externe affichée dans la fenêtre (qui peut être exploité par les autres routines du plug-in 4D).

Les six premiers paramètres sont identiques à ceux de la commande Creer fenetre. Cependant, aucun d'entre eux n'est optionnel.

Creer fenetre externe crée une fenêtre "sans mode", c'est-à-dire que la commande n'attend pas d'action de l'utilisateur, ce qui vous permet d'afficher plusieurs fenêtres actives simultanément. Vous pouvez naviguer parmi chaque fenêtre en cliquant dessus. Vous pouvez modifier celle qui se trouve au premier plan. Si le type de la fenêtre comporte une barre de titre, une case du menu Système (Windows) ou une case de fermeture (Macintosh) sera ajoutée à la fenêtre pour permettre à l'utilisateur de refermer la fenêtre.

Exemples

(1) L'exemple suivant ouvre une fenêtre externe et affiche une zone externe 4D Write :

```
⇒ wrWind:=Creer fenetre externe (50; 50; 350; 450; 8; "Ecrire lettre"; "_4D WRITE")
```

(2) L'exemple suivant referme la fenêtre externe ouverte dans l'exemple précédent :

```
FERMER FENETRE (wrWind)
```

Référence

Creer fenetre, FERMER FENETRE.

FERMER FENETRE {(numFenêtre)}

Paramètre	Type		Description
numFenêtre	RefFen	→	Numéro de référence de fenêtre externe ou Fenêtre de premier plan du process si ce paramètre est omis

Description

FERMER FENETRE referme la dernière fenêtre créée à l'aide de la commande Creer fenetre dans le process courant. S'il n'y a pas de fenêtre personnalisée ouverte, FERMER FENETRE ne fait rien ; la commande ne ferme pas les fenêtres standard. Si FERMER FENETRE est appelée alors qu'un formulaire est actif dans la fenêtre, elle n'a pas d'effet non plus. Vous devez appeler FERMER FENETRE lorsque vous avez fini d'utiliser une fenêtre ouverte avec Creer fenetre.

Il est inutile de passer un numéro à FERMER FENETRE lorsque vous l'utilisez pour refermer de fenêtres ouvertes à l'aide de la fonction Creer fenetre. En effet, si plusieurs fenêtres ont été ouvertes par une succession d'appels à Creer fenetre, elles ne pourront être refermées que dans l'ordre inverse de leur création.

Si vous passez en paramètre la référence d'une zone externe créée à l'aide de la fonction Creer fenetre externe, FERMER FENETRE referme la fenêtre externe. Pour plus d' informations sur les fenêtres externes, reportez-vous à la description de la fonction Creer fenetre externe.

Exemple

L'exemple suivant ouvre une fenêtre et crée des enregistrements à l'aide de la commande AJOUTER ENREGISTREMENT. Une fois les enregistrements ajoutés, la fenêtre est fermée par la commande FERMER FENETRE :

```
Creer fenetre (5; 40; 250; 300; 0; "Nouvel employé")
Repeter
   AJOUTER ENREGISTREMENT ([Employés]) ` Ajout d'un enregistrement d'employé
Jusque (OK = 0) ` Boucle jusqu'à ce que l'utilisateur annule
⇒  FERMER FENETRE ` Fermeture de la fenêtre
```

Référence

Creer fenetre, Creer fenetre externe.

EFFACER FENETRE {(fenêtre)}

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de fenêtre ou Fenêtre au premier plan du process courant si ce paramètre est omis

Description

La commande EFFACER FENETRE efface le contenu de la fenêtre dont vous avez passé la référence dans fenêtre.

Si vous omettez le paramètre fenêtre, EFFACER FENETRE efface le contenu de la fenêtre de premier plan du process courant.

Généralement, vous utiliserez EFFACER FENETRE en combinaison avec MESSAGE et POSITION MESSAGE. Dans ce cas, EFFACER FENETRE efface le contenu de la fenêtre et place le curseur dans son angle supérieur gauche, c'est-à-dire à la position correspondant à POSITION MESSAGE (0; 0).

Ne confondez pas EFFACER FENETRE, qui efface le contenu d'une fenêtre, et FERMER FENETRE, qui supprime la fenêtre de l'écran.

Référence

MESSAGE, POSITION MESSAGE.

REDESSINER FENETRE {(fenêtre)}

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de la fenêtre ou Fenêtre de premier plan du process courant si ce paramètre est omis

Description

La commande REDESSINER FENETRE provoque une mise à jour du contenu de la fenêtre dont le numéro de référence est passé dans fenêtre.

Si vous omettez le paramètre fenêtre, REDESSINER FENETRE s'appliquera à la fenêtre de premier plan du process courant.

Note : 4e Dimension gère à votre place les mises à jour graphiques des fenêtres à chaque fois que vous déplacez, redimensionnez ou passez au premier plan une fenêtre ainsi qu'à chaque fois que vous changez le formulaire et/ou les valeurs affiché(e)s dans une fenêtre. Cette commande est rarement utilisée.

Référence

EFFACER FENETRE.

DEPLACER FENETRE

Paramètre	Type	Description

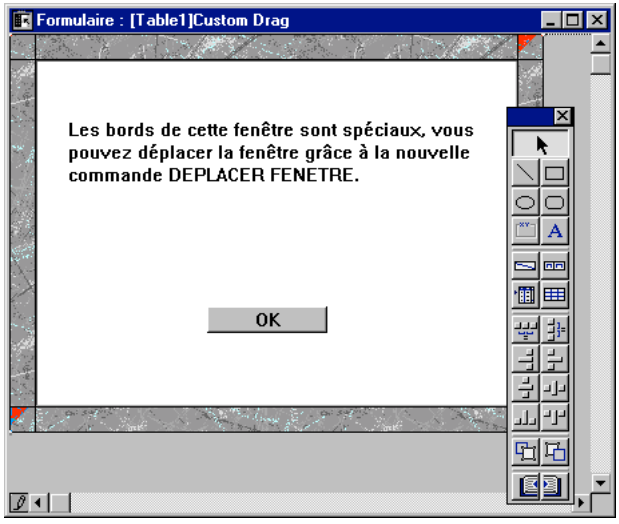
Cette commande ne requiert pas de paramètre

Description

La commande DEPLACER FENETRE fait glisser la fenêtre de premier plan courante en suivant les mouvements de la souris. Généralement, cette commande est appelée depuis la méthode d'un objet capable de répondre instantanément aux clics souris (par exemple un bouton invisible).

Exemple

Le formulaire suivant, présenté ici en mode Structure, contient un fond créé par une image statique par-dessus laquelle quatre boutons invisibles ont été placés (un par côté) :



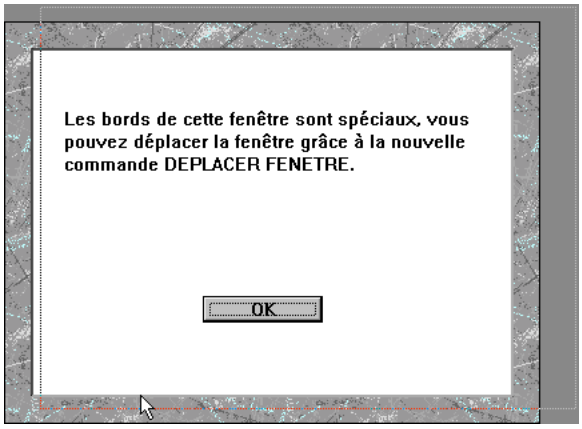
Chaque bouton est associé à la méthode suivante :

DEPLACER FENETRE ` Commencer à faire glisser la fenêtre au premier clic

En mode Utilisation ou Menus créés, après l'exécution de la méthode projet suivante :

```
Creer fenetre(50;50;50+400;50+300;2)
DIALOGUE([Table1];"Custom Drag")
FERMER FENETRE
```

... vous obtenez une fenêtre semblable à celle-ci :



Vous pouvez la déplacer en cliquant sur les bordures.

Référence

CHANGER COORDONNEES FENETRE, COORDONNEES FENETRE.

CACHER FENETRE {(fenêtre)}

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de la fenêtre ou Fenêtre de premier plan du process courant si omis

Description

La commande CACHER FENETRE permet de masquer la fenêtre dont vous avez passé le numéro de référence dans fenêtre ou, si ce paramètre est omis, la fenêtre de premier plan du process courant. Cette commande vous permet, par exemple, dans un process comportant plusieurs fenêtres, de ne conserver à l'écran que la fenêtre active.

La fenêtre disparaît de l'écran mais reste ouverte. Vous pouvez continuer à lui appliquer par programmation tout traitement supporté par les fenêtres 4D.

Pour réafficher une fenêtre masquée par CACHER FENETRE :
• Utilisez la commande AFFICHER FENETRE et passez-lui le numéro de référence de la fenêtre.
• Ou bien, dans la page **Process** de l'Explorateur d'exécution, sélectionnez le process dans lequel la fenêtre est gérée (process de gestion de la fenêtre ou Process principal) puis cliquez sur le bouton **Montrer**.

Pour cacher toutes les fenêtres d'un process, utilisez la commande CACHER PROCESS.

Exemple

Cet exemple est la méthode d'un bouton placé dans un formulaire entrée. Ce bouton ouvre une boîte de dialogue dans une nouvelle fenêtre du même process. Vous souhaitez masquer les autres fenêtres du process (un formulaire de saisie et une palette d'outils) afin de ne présenter que la boîte de dialogue. Une fois que celle-ci a été validée, vous réaffichez les fenêtres du process.

```
      ` Méthode objet du bouton "Informations"
⇒     CACHER FENETRE(Saisie) ` Masquer la fenêtre de saisie
⇒     CACHER FENETRE(Palette) ` Masquer la palette
      $Infos:=Creer fenetre(20;100;500;400;8) ` Créer la fenêtre d'informations
      ... ` Placer ici les instructions nécessaires au remplissage et à la gestion du dialogue
      FERMER FENETRE($Infos) ` Fermer le dialogue
⇒     AFFICHER FENETRE(Saisie)
⇒     AFFICHER FENETRE(Palette) ` Réafficher les autres fenêtres du process
```

Référence

AFFICHER FENETRE.

AFFICHER FENETRE {(fenêtre)}

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de la fenêtre ou Fenêtre de premier plan du process courant si omis

Description

La commande AFFICHER FENETRE permet d'afficher la fenêtre dont vous avez passé le numéro de référence dans fenêtre ou, si ce paramètre est omis, la fenêtre de premier plan du process courant.

La fenêtre doit au préalable avoir été cachée à l'aide de la commande CACHER FENETRE. Si la fenêtre est déjà affichée, la commande ne fait rien.

Exemple

Voir l'exemple de la commande CACHER FENETRE.

Référence

CACHER FENETRE.

MAXIMISER FENETRE {(fenêtre)}

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de la fenêtre Si omis = Toutes les fenêtres (Windows) ou Fenêtre de premier plan du process courant (MacOS)

Description

La commande MAXIMISER FENETRE provoque le zoom de la fenêtre dont vous avez passé le numéro de référence dans fenêtre ou, si ce paramètre est omis, de toutes les fenêtres de l'application (sous Windows) ou de la fenêtre de premier plan du process courant (sous MacOS).

Cette commande produit le même effet qu'un clic sur la case de zoom d'une fenêtre de l'application 4D :

Sous Windows
La fenêtre est agrandie et s'adapte à la taille courante de la fenêtre de l'application. Si vous ne passez pas le paramètre fenêtre, toutes les fenêtres de l'application sont maximisées. La fenêtre maximisée est passée au premier plan.


Case de zoom ("bouton d'agrandissement") sous Windows

Sous MacOS
La fenêtre est agrandie et s'adapte à la taille de l'écran principal. Si vous ne passez pas le paramètre fenêtre, la fenêtre du premier plan du process courant est maximisée.


Case de zoom sous MacOS

Les fenêtres que vous souhaitez maximiser doivent comporter une case de zoom. Si le type de fenêtre n'en contient pas, la commande ne fait rien (pour plus d'informations sur ce point, reportez-vous à la section Types de fenêtres).

La fenêtre est dimensionnée à sa taille "maximale". Si la fenêtre est un formulaire dont la taille maximale a été définie dans les Propriétés du formulaire, le zoom de la fenêtre se limitera à cette taille.

Si fenêtre est déjà maximisée, la commande ne fait rien.

Un clic ultérieur sur la case de zoom ou l'appel de la commande MINIMISER FENETRE provoque le retour de la fenêtre à sa taille initiale.
Sous Windows, un clic sur la case de zoom ou l'appel de la commande MINIMISER FENETRE (sans paramètre) entraîne le retour à leur taille initiale de toutes les fenêtres de l'application.

Exemple

Vous souhaitez que votre base de données s'ouvre sur une fenêtre "plein écran". Pour cela, vous placez la commande MAXIMISER FENETRE dans la Méthode base Sur ouverture :

```
` Méthode base Sur ouverture

⇒ MAXIMISER FENETRE
```

Référence

MINIMISER FENETRE.

MINIMISER FENETRE {(fenêtre)}

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de la fenêtre Si omis = Toutes les fenêtres (Windows) ou Fenêtre de premier plan du process courant (MacOS)

Description

La commande MINIMISER FENETRE provoque un zoom arrière de la fenêtre dont vous avez passé le numéro de référence dans fenêtre ou, si ce paramètre est omis, de toutes les fenêtres de l'application (sous Windows) ou de la fenêtre de premier plan du process courant (sous MacOS).

Cette commande produit le même effet qu'un clic sur la case de réduction d'une fenêtre de l'application 4D ayant été préalablement maximisée :

Sous Windows
La fenêtre est réduite et retrouve sa taille initiale. Si vous ne passez pas le paramètre fenêtre, toutes les fenêtres de l'application sont redimensionnées à leur taille initiale.

Case de réduction sous Windows

Sous MacOS
La fenêtre est réduite et retrouve sa taille initiale. Si vous ne passez pas le paramètre fenêtre, la fenêtre de premier plan du process courant est minimisée.

Case de zoom/réduction sous MacOS

Si la ou les fenêtres concernées n'ont pas été préalablement maximisées (manuellement ou à l'aide de MAXIMISER FENETRE), la commande ne fait rien. De même, si le type de fenêtre ne comporte pas de case de zoom, la commande ne fait rien (pour plus d'informations sur ce point, reportez-vous à la section Types de fenêtres).

Note : Sous Windows, ne confondez pas cette fonction avec la réduction de la fenêtre sous forme d'icône, accessible par l'intermédiaire du bouton suivant :

Référence

MAXIMISER FENETRE.

Titre fenetre

version 6.0

Titre fenetre {(fenêtre)} → Chaîne

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de la fenêtre ou Fenêtre de premier plan du process courant si ce paramètre est omis
Résultat	Chaîne	←	Titre de la fenêtre

Description

La commande Titre fenetre retourne le titre de la fenêtre dont le numéro de référence est passé dans fenêtre. Si la fenêtre n'existe pas, une chaîne vide est retournée.

Si vous omettez le paramètre fenêtre, Titre fenetre retourne le titre de la fenêtre de premier plan du process courant.

Exemple

Reportez-vous à l'exemple de la commande CHANGER TITRE FENETRE.

Référence

CHANGER TITRE FENETRE.

CHANGER TITRE FENETRE (titre{; fenêtre})

Paramètre	Type		Description
titre	Alpha	→	Titre de la fenêtre
fenêtre	RefFen	→	Numéro de référence de fenêtre ou Fenêtre au premier plan du process courant si ce paramètre est omis

Description

La commande CHANGER TITRE FENETRE remplace le titre de la fenêtre dont le numéro de référence est passé dans fenêtre par le texte passé dans titre (longueur maximale 80 caractères).

Si la fenêtre n'existe pas, CHANGER TITRE FENETRE ne fait rien.

Si vous omettez le paramètre fenêtre, CHANGER TITRE FENETRE remplace le titre de la fenêtre de premier plan du process courant.

Note : En mode Utilisation, 4e Dimension définit automatiquement les titres des fenêtres — par exemple “Saisie pour table1” est affiché lorsque vous passez en saisie de données. Si vous changez le titre d'une fenêtre du mode Utilisation, il est probable que 4D le remplacera par la suite. En revanche, en mode Menus créés, 4e Dimension ne modifie pas le titre des fenêtres.

Exemple

Vous effectuez une saisie dans un formulaire et vous cliquez sur un bouton qui déclenche une longue opération (par exemple une modification par programmation des enregistrements liés affichés dans un sous-formulaire). Vous pouvez afficher des informations sur la progression des opérations dans le titre de la fenêtre :

```
   ` Méthode objet du bouton bAnalyse
Au cas ou
   : (Evenement formulaire=Sur clic souris)
         ` Sauvegarde du titre courant de la fenêtre dans une variable
      $vsTitreCour:=Titre fenetre
         ` Commencer l'opération longue
      DEBUT SELECTION([Lignes facture])
      Boucle($vlRecord;1;Enregistrements trouves([Lignes facture]))
         FAIRE QUELQUE CHOSE
            ` Afficher la progression
         CHANGER TITRE FENETRE("Traitement de la ligne #"+Chaine($vlEnreg))
      Fin de si
         ` Remettre en place l'ancien titre de fenêtre
      CHANGER TITRE FENETRE($vsTitreCour)
Fin de cas
```

Référence

Titre fenetre.

CACHER BARRE OUTILS

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

La commande CACHER BARRE OUTILS rend invisible la barre d'outils. Si la barre d'outils était déjà cachée, CACHER BARRE OUTILS ne fait rien.

Référence

AFFICHER BARRE DE MENUS, AFFICHER BARRE OUTILS, CACHER BARRE DE MENUS.

AFFICHER BARRE OUTILS

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande AFFICHER BARRE OUTILS rend visible la barre d'outils. Si la barre d'outils était déjà visible, AFFICHER BARRE OUTILS ne fait rien.

Référence

AFFICHER BARRE DE MENUS, CACHER BARRE DE MENUS, CACHER BARRE OUTILS.

LISTE FENETRES (fenêtres{; *})

Paramètre	Type		Description
fenêtres	Tableau	←	Tableau des numéros de référence des fenêtres
*	*	→	Si omis, ignorer fenêtres flottantes ; si spécifié, tenir compte des fenêtres flottantes

Description

La commande LISTE FENETRES remplit le tableau fenêtres avec les numéros de référence des fenêtres ouvertes dans tous les process (process moteur et process utilisateur).
Si vous ne passez pas le paramètre optionnel *, les fenêtres flottantes sont ignorées.

Exemple

La méthode projet suivante place en "mosaïque" toutes les fenêtres ouvertes (à l'exception des fenêtres flottantes et des boîtes de dialogue) :

```
      ` Méthode projet Mosaïque
⇒     LISTE FENETRES($alWnd)
      $vlLeft:=10
      $vlTop:=80 ` Laissons de la place à la barre d'outils
      Boucle ($vlWnd;1;Taille tableau($alWnd))
         Si (Type fenetre($alWnd{$vlWnd}) # Fenêtre modale)
            COORDONNEES FENETRE($vlWL;$vlWT;$vlWR;$vlWB;$alWnd{$vlWnd})
            $vlWR:=$vlLeft+($vlWR-$vlWL)
            $vlWB:=$vlTop+($vlWB-$vlWT)
            $vlWL:=$vlLeft
            $vlWT:=$vlTop
            CHANGER COORDONNEES FENETRE($vlWL;$vlWT;$vlWR;$vlWB;
                                                $alWnd{$vlWnd})
            $vlLeft:=$vlLeft+10
            $vlTop:=$vlTop+25
         Fin de si
      Fin de boucle
```

Note : Cette méthode pourrait être améliorée par l'ajout de tests sur la taille de la fenêtre principale (sous Windows) et l'emplacement du ou des écran(s) (sous MacOS).

Référence

Process de la fenetre, Type fenetre.

Type fenetre

version 6.0

Type fenetre {(fenêtre)}

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de la fenêtre ou Fenêtre de premier plan du process courant si ce paramètre est omis

Description

La commande Type fenetre retourne le type de fenêtre 4e Dimension dont vous avez passé la référence dans fenêtre. Si la fenêtre n'existe pas, Type fenetre retourne 0 (zéro).

Sinon, Type fenetre retourne une des constantes prédéfinies suivantes :

Constante	Type	Valeur
Fenêtre standard	Entier long	8
Fenêtre modale	Entier long	9
Fenêtre externe	Entier long	5
Fenêtre flottante	Entier long	14

Si vous omettez le paramètre fenêtre, Type fenetre s'applique à la fenêtre de premier plan du process courant.

Exemple

Reportez-vous à l'exemple de la commande LISTE FENETRES.

Référence

COORDONNEES FENETRE, Process de la fenetre, Titre fenetre.

Process de la fenetre

version 6.0

Process de la fenetre {(fenêtre)} → Numérique

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de fenêtre
Résultat	Numérique	←	Numéro de référence de process

Description

La commande Process de la fenetre retourne le numéro du process qui exécute la fenêtre dont le numéro de référence est passé dans fenêtre. Si la fenêtre n'existe pas, la commande retourne 0 (zéro).

Si vous omettez le paramètre fenêtre, Process de la fenetre retourne le numéro du process de la fenêtre de premier plan du process courant.

Référence

Numero du process courant.

COORDONNEES FENETRE (gauche; haut; droite; bas{; fenêtre})

Paramètre	Type		Description
gauche	Numérique	←	Coordonnée gauche de l'intérieur de la fenêtre
haut	Numérique	←	Coordonnée supérieure de l'intérieur de la fenêtre
droite	Numérique	←	Coordonnée droite de l'intérieur de la fenêtre
bas	Numérique	←	Coordonnée inférieure de l'intérieur de la fenêtre
fenêtre	RefFen	→	Numéro de référence de la fenêtre ou Fenêtre de premier plan du process si ce paramètre est omis

Description

La commande COORDONNEES FENETRE retourne les coordonnées globales de la fenêtre dont vous avez passé le numéro de référence dans le paramètre fenêtre. Si la fenêtre n'existe pas, les variables des paramètres sont inchangées.

Si vous omettez le paramètre fenêtre, COORDONNEES FENETRE s'applique à la fenêtre de premier plan du process courant.

Ces coordonnées sont exprimées relativement au coin supérieur gauche de la zone de contenu de la fenêtre d'application (sous Windows) ou de l'écran principal (sous MacOS). Les coordonnées retournent le rectangle correspondant à la zone de contenu de la fenêtre (en excluant les barres de titres et les bordures).

Exemple

Reportez-vous à l'exemple de la commande LISTE FENETRES.

Référence

CHANGER COORDONNEES FENETRE.

CHANGER COORDONNEES FENETRE

version 6.0

CHANGER COORDONNEES FENETRE (gauche; haut; droite; bas{; fenêtre})

Paramètre	Type		Description
gauche	Numérique	→	Coordonnée gauche de l'intérieur de la fenêtre
haut	Numérique	→	Coordonnée supérieure de l'intérieur de la fenêtre
droite	Numérique	→	Coordonnée droite de l'intérieur de la fenêtre
bas	Numérique	→	Coordonnée inférieure de l'intérieur de la fenêtre
fenêtre	RefFen	→	Numéro de référence de la fenêtre ou Fenêtre de premier plan du process si ce paramètre est omis

Description

La commande CHANGER COORDONNEES FENETRE modifie les coordonnées globales de la fenêtre dont vous avez passé le numéro de référence dans le paramètre fenêtre. Si la fenêtre n'existe pas, la commande ne fait rien.

Si vous omettez le paramètre fenêtre, CHANGER COORDONNEES FENETRE s'applique à la fenêtre de premier plan du process courant.

Ces coordonnées doivent être exprimées relativement au coin supérieur gauche de la zone de contenu de la fenêtre d'application (sous Windows) ou de l'écran principal (sous MacOS). Les coordonnées décrivent le rectangle correspondant à la zone de contenu de la fenêtre (en excluant les barres de titres et les bordures).

Attention : Utilisez cette commande avec précaution, car vous pouvez déplacer une fenêtre en-dehors des limites de la fenêtre principale (sous Windows) ou de l'écran (sous MacOS). Pour éviter cela, vous pouvez utiliser des fonctions telles que Largeur ecran et Hauteur ecran pour bien vérifier les nouvelles coordonnées de la fenêtre.

Exemple

Reportez-vous à l'exemple de la commande LISTE FENETRES.

Référence

COORDONNEES FENETRE, DEPLACER FENETRE.

Fenetre premier plan

version 6.0

Fenetre premier plan {(*)} → RefFen

Paramètre	Type		Description
*	*	→	Si omis = ignorer les fenêtres flottantes Si spécifié = prendre en compte les fenêtres flottantes
Résultat	RefFen	←	Numéro de référence de fenêtre

Description

La commande Fenetre premier plan retourne le numéro de référence de la fenêtre actuellement située au premier plan.

Référence

Fenetre suivante, Process de premier plan.

Fenetre suivante

version 6.0

Fenetre suivante (fenêtre) → Numérique

Paramètre	Type		Description
fenêtre	RefFen	→	Numéro de référence de la fenêtre
Résultat	Numérique	←	Numéro de référence de fenêtre

Description

La commande Fenetre suivante retourne le numéro de référence de la fenêtre située “derrière” la fenêtre dont vous avez passé le numéro de référence dans fenêtre (en fonction de l'ordre des fenêtres).

Référence

Fenetre premier plan.

Chercher fenetre (gauche; haut{; partieFenêtre}) → RefFenêtre

Paramètre	Type		Description
gauche	Numérique	→	Coordonnée globale gauche
haut	Numérique	→	Coordonnée globale supérieure
partieFenêtre	Numérique	←	Numéro de partie de fenêtre
Résultat	RefFenêtre	←	Numéro de référence de fenêtre

Description

La commande Chercher fenetre retourne (s'il existe) le numéro de référence de la première fenêtre "touchée" par le point dont vous passez les coordonnées dans gauche et haut.

Ces coordonnées doivent être exprimées relativement au coin supérieur gauche de la zone de contenu (l'intérieur) de la fenêtre d'application (sous Windows) ou de l'écran principal (sous MacOS).

Si vous passez le paramètre partieFenêtre, qu'une fenêtre ait été trouvée ou non, ce paramètre retournera une des constantes prédéfinies suivantes :

Constantes	Type	Valeur	Plate-forme
Dans barre de menu	Entier long	1	Macintosh
Dans fenêtre système	Entier long	2	Macintosh
Dans zone contenu	Entier long	3	Windows ou Macintosh
Dans barre de titre	Entier long	4	Macintosh
Dans case de contrôle de taille	Entier long	5	Macintosh
Dans case de fermeture	Entier long	6	Macintosh
Dans case de zoom	Entier long	8	Macintosh

Référence

Fenetre premier plan, Fenetre suivante.

Creer fenetre formulaire

version 6.5

Creer fenetre formulaire ({table; }nomForm{; type{; posH{; posV{; *}}}}) → RefFen

Paramètre	Type		Description
table	Table	→	Table du formulaire ou Table par défaut si ce paramètre est omis
nomForm	Alpha	→	Nom du formulaire
type	Entier long	→	Type de la fenêtre
posH	Entier long	→	Position horizontale de la fenêtre
posV	Entier long	→	Position verticale de la fenêtre
*	*	→	Conserver les position et taille précédentes de la fenêtre
Résultat	RefFen	←	Numéro de référence de la fenêtre

Description

La commande Creer fenetre formulaire crée une nouvelle fenêtre utilisant les propriétés de taille et de redimensionnement du formulaire nomForm, passé en paramètre.

A noter que nomForm n'est pas affiché dans la fenêtre créée. Il vous appartient, si vous le souhaitez, d'afficher le formulaire (par exemple à l'aide de la commande AJOUTER ENREGISTREMENT).

Par défaut (si le paramètre type n'est pas passé), la fenêtre créée est de type standard et comporte une case de fermeture. A la différence de la commande Creer fenetre, aucune méthode n'est associée à cette case de fermeture : un clic sur la case de fermeture provoquera simplement l'annulation du formulaire, sauf si l'événement Sur case de fermeture est activé pour le formulaire, auquel cas le code associé à cet événement sera exécuté.

Si le formulaire nomForm est redimensionnable, la fenêtre créée comporte également une case de zoom et une case de contrôle de taille.

Note : Vous pouvez connaître les principales propriétés d'un formulaire à l'aide de la commande LIRE PROPRIETES FORMULAIRE.

Le paramètre optionnel type vous permet de spécifier un type de fenêtre. Ce paramètre doit contenir une des constantes prédéfinies suivantes (placées dans le thème "Creer fenetre") :

Constante	Type	Valeur
Fenêtre standard	Entier long	8
Dialogue modal	Entier long	1
Dialogue modal déplaçable	Entier long	5
Fenêtre palette	Entier long	720

Le paramètre optionnel posH vous permet de définir l'emplacement horizontal de la fenêtre. Vous pouvez passer dans ce paramètre une coordonnée fixe exprimée en points (cf. commande Creer fenetre), ou l'une des constantes prédéfines suivantes (placées dans le thème "Position fenetre") :

Constante	Type	Valeur
Centrée horizontalement	Entier long	65536
A gauche	Entier long	131072
A droite	Entier long	196608

Le paramètre optionnel posV vous permet de définir l'emplacement vertical de la fenêtre. Vous pouvez passer dans ce paramètre une coordonnée fixe exprimée en points (cf. commande Creer fenetre), ou l'une des constantes prédéfines suivantes (placées dans le thème "Position fenetre") :

Constante	Type	Valeur
Centrée verticalement	Entier long	262144
En haut	Entier long	327680
En bas	Entier long	393216

Ces paramètres tiennent compte de la présence de la barre d'outils, de la barre de menus, et de la taille courante de la fenêtre de l'application (sous Windows).

Si vous passez le paramètre optionnel *, les position et taille courantes de la fenêtre sont mémorisées au moment où elle est refermée. Lorsque la fenêtre est réouverte par la suite, elle conserve sa position et sa taille précédentes. Dans ce cas, les paramètres posV et posH ne sont utilisés que pour la première ouverture de la fenêtre.

Exemples

(1) L'instruction suivante ouvre une fenêtre standard avec case de fermeture automatiquement ajustée à la taille du formulaire "Entrée". Comme le formulaire a la propriété "Redimensionnable", la fenêtre comporte également une case de contrôle de taille et une case de zoom :

⇒ $refFen:=**Creer fenetre formulaire** ([Table1];"Entrée")

(2) L'instruction suivante ouvre, en haut et à gauche de l'écran, une palette flottante. Cette palette conservera sa précédente position à chaque nouvelle ouverture :

⇒ $refFen:=**Creer fenetre formulaire** ([Table1]; "Outils"; Fenêtre palette; A gauche; En haut;*)

Référence

Creer fenetre, LIRE PROPRIETES FORMULAIRE.

LIRE PROPRIETES FORMULAIRE ({table; }nomForm; largeur; hauteur{; nbPages{; largeurFixe{; hauteurFixe{; titre}}}})

Paramètre	Type		Description
table	Table	→	Table du formulaire ou Table par défaut si ce paramètre est omis
nomForm	Alpha	→	Nom du formulaire
largeur	Entier long	←	Largeur du formulaire (en pixels)
hauteur	Entier long	←	Hauteur du formulaire (en pixels)
nbPages	Entier long	←	Nombre de pages du formulaire
largeurFixe	Booléen	←	Vrai = Largeur fixe, Faux = Largeur variable
hauteurFixe	Booléen	←	Vrai = Hauteur fixe, Faux = Hauteur variable
titre	Texte	←	Nom de la fenêtre du formulaire

Description

La commande LIRE PROPRIETES FORMULAIRE retourne des propriétés du formulaire nomForm.

Les paramètres largeur et hauteur retournent (en pixels) la largeur et la hauteur du formulaire. Ces valeurs sont déterminées à partir des propriétés de dimensionnement du formulaire :

• Si la taille du formulaire est **automatique**, sa largeur et sa hauteur sont calculées de manière à ce qu'il affiche tous les objets qu'il contient, en tenant compte, le cas échéant, des marges horizontale et verticale qui ont été définies.

• Si la taille du formulaire est **fixe**, sa largeur et sa hauteur sont celles qui ont été saisies manuellement dans les zones correspondantes.

• Si la taille du formulaire est **basée sur un objet**, sa largeur et sa hauteur sont calculées par rapport à la position de cet objet.

Le paramètre nbPages retourne le nombre de pages du formulaire, page 0 (zéro) non comprise.

Les paramètres largeurFixe et hauteurFixe indiquent si la largeur et la hauteur du formulaire sont fixes (le paramètre contient Vrai) ou redimensionnables (le paramètre contient Faux).

Le paramètre titre retourne le nom de la fenêtre du formulaire, tel qu'il a été défini. Si aucun nom n'a été défini, le paramètre titre contient une chaîne vide.

Référence

Creer fenetre formulaire.

20

Fonctions mathématiques

Abs (nombre) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Nombre dont vous voulez obtenir la valeur absolue
Résultat	Numérique	←	Valeur absolue de nombre

Description

Abs retourne la valeur absolue (positive et sans signe) de nombre. Si nombre est négatif, sa valeur positive est retournée. Si nombre est positif, il est retourné inchangé.

Exemple

L'exemple suivant retourne la valeur absolue de -10,3, qui est 10,3 :

```
⇒ vVector := Abs (–10,3)
```

Ent (nombre) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Valeur dont vous voulez obtenir la partie entière
Résultat	Numérique	←	Partie entière de nombre

Description

Ent retourne la partie entière de nombre en l'arrondissant à l'entier inférieur.

Exemple

L'exemple suivant illustre le fonctionnement de Ent pour les nombres positifs et négatifs. Notez que la partie décimale du nombre est supprimée :

```
⇒ x := Ent (123,4) ` x vaut 123
⇒ y := Ent (–123,4) ` y vaut –124
```

Référence

Dec.

Dec (nombre) → Numérique

paramètre	Type		Description
nombre	Numérique	→	Valeur dont voulez obtenir la partie décimale
Résultat	Numérique	←	Partie décimale de nombre

Description

Dec retourne la partie décimale de nombre. La valeur retournée est toujours positive ou nulle.

Exemple

L'exemple suivant utilise une valeur monétaire exprimée sous forme numérique et en extrait les parties "francs" et "centimes". Si vlMontant valait 7,31, vlFrancs vaudrait 7 et vlCentimes 31 :

```
      vlFrancs := Ent (vlMontant) ` Extraire les francs
⇒     vlCentimes := Dec (vlMontant) * 100 ` Extraire la partie décimale
```

Référence

Ent.

Arrondi (nombre; nbDécimales) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Nombre à arrondir
nbDécimales	Numérique	→	Nombre de décimales de l'arrondi
Résultat	Numérique	←	Valeur de nombre arrondie avec une précision égale à nbDécimales

Description

Arrondi retourne la valeur arrondie de nombre avec une précision égale à nbDécimales.

Si nbDécimales est positif, l'arrondi se fait sur la partie décimale de nombre. Si nbDécimales est négatif, l'arrondi se fait sur la partie entière de nombre.

Si le chiffre placé derrière le nombre de décimales défini par nbDécimales est compris entre 5 et 9, nombre est arrondi à la valeur supérieure s'il est positif et inférieure s'il est négatif. Si le chiffre placé derrière la dernière décimale est compris entre 0 et 4, la fonction arrondit nombre vers zéro.

Exemples

L'exemple suivant illustre la manière dont Arrondi fonctionne dans différents cas. A chaque ligne, une valeur est assignée à la variable vRésultat. Les commentaires décrivent le résultat :

```
⇒ vRésultat := Arrondi (16,857; 2)   ` vRésultat vaut 16,86
⇒ vRésultat := Arrondi (32345,67; –3) ` vRésultat vaut 32000
⇒ vRésultat := Arrondi (29,8725; 3)  ` vRésultat vaut 29,873
⇒ vRésultat := Arrondi (–1,5; 0)   ` vRésultat vaut –2
```

Référence

Troncature.

Troncature (nombre; nbDécimales) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Nombre à tronquer
nbDécimales	Numérique	→	Nombre de décimales à conserver
Résultat	Numérique	←	nombre tronqué à partir du nombre de décimales indiqué par nbDécimales

Description

Troncature retourne nombre dont la partie décimale a été tronquée à partir du nombre de décimales spécifié par nbDécimales. Troncature arrondit toujours nombre à la valeur inférieure.

Si nbDécimales est positif, la troncature se fait sur la partie décimale de nombre. Si nbDécimales est négatif, la troncature se fait sur la partie entière de nombre.

Exemples

L'exemple suivant illustre la manière dont Troncature fonctionne dans différents cas. A chaque ligne, une valeur est assignée à la variable vRésultat. Les commentaires décrivent le résultat :

```
⇒    vRésultat := Troncature (216,897; 1)    ` Résultat prend la valeur 216,8
⇒    vRésultat := Troncature (216,897; –1)   ` Résultat prend la valeur 210
⇒    vRésultat := Troncature (–216,897; 1)   ` Résultat prend la valeur –216,9
⇒    vRésultat := Troncature (–216,897; –1)  ` Résultat prend la valeur –220
```

Référence

Arrondi.

Hasard → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Nombre aléatoire

Description

Hasard retourne une valeur entière aléatoire comprise entre 0 et 32 767 (inclus).

Pour que la valeur aléatoire soit située dans un intervalle donné, utilisez la formule suivante :

```
(Hasard % (fin – début + 1)) + début
```

La valeur début est le premier nombre de l'intervalle, fin est le dernier.

Exemple

L'exemple suivant assigne une valeur entière aléatoire entre 10 et 30 à la variable vRésultat :

```
⇒ Résultat := (Hasard % 21) + 10
```

Modulo (nombre1; nombre2) → Numérique

Paramètre	Type		Description
nombre1	Numérique	→	Nombre à diviser (numérateur)
nombre2	Numérique	→	Nombre diviseur (dénominateur)
Résultat	Numérique	←	Reste de la division entière de nombre1 par nombre2

Description

La fonction Modulo divise nombre1 par nombre2 et retourne le reste sous forme d'un nombre entier.

Note : Modulo accepte des expressions de type Entier, Entier long et Réel (numérique). Cependant, si nombre1 et/ou nombre2 sont des nombres réels, ils sont arrondis avant le calcul du Modulo.

Vous pouvez également utiliser l'opérateur "%" pour calculer le reste d'une division (reportez-vous à la section Opérateurs numériques).

ATTENTION : L'opérateur % retourne des résultats valides uniquement avec des expressions de type Entier et Entier long. Si vous voulez calculer le modulo de nombres réels, vous devez utiliser la commande Modulo.

Exemple

L'exemple suivant illustre le fonctionnement de Modulo dans différents cas de figure. A chaque ligne, un nombre est assigné à la variable vRésultat. Les commentaires fournissent le résultat obtenu :

```
⇒ vRésultat := Modulo(3;2)   ` vRésultat prend la valeur 1
⇒ vRésultat := Modulo(4;2)   ` vRésultat prend la valeur 0
⇒ vRésultat := Modulo(3.5;2)   ` vRésultat prend la valeur 0
```

Référence

Opérateurs numériques.

Racine carree (nombre) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Nombre duquel calculer la racine carrée
Résultat	Numérique	←	Racine carrée de nombre

Description

Racine carree retourne la racine carrée de nombre.

Exemples

(1) La ligne :

```
⇒ $vrRacineDeDeux := Racine carree (2)
```

affecte la valeur 1,414213562373 à la variable $vrRacineDeDeux.

(2) La méthode listée ci-dessous retourne l'hypoténuse du triangle rectangle dont les deux côtés sont passés en paramètres :

```
      ` Méthode Hypoténuse
      ` Hypoténuse ( Réel ; Réel ) -> Réel
      ` Hypoténuse ( côtéA ; côtéB ) -> Hypoténuse
C_REEL($0;$1;$2)
⇒ $0 := Racine carree(($1^2)+($2^2))
```

Par exemple, Hypoténuse (4;3) retourne 5.

Référence

Opérateurs numériques.

Log (nombre) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Nombre dont vous voulez obtenir le logarithme népérien
Résultat	Numérique	←	Logarithme népérien de nombre

Description

Log retourne le logarithme népérien de nombre. Log est la fonction inverse de Exp.

Note : 4D fournit la constante prédéfinie Nombre e (2,71828...).

Exemple

L'exemple suivant affiche 1 :

```
⇒ ALERTE (Chaine(Log(Exp (1)))
```

Référence

Exp.

Exp (nombre) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Nombre à évaluer
Résultat	Numérique	←	Exponentielle de nombre

Description

Exp retourne l'exponentielle (e=2,71828...) de nombre. Exp est la fonction inverse de Log.

Note : La fonction exponentielle, qui au nombre réel x fait correspondre le nombre réel y, est notée y = ex. 4D fournit la constante prédéfinie Nombre e (2,71828...).

Exemple

L'exemple suivant assigne l'exponentielle de 1 à vrE (le logarithme de vrE est 1) :

```
⇒ vrE := Exp (1)  ` vrE vaut e (e = 2,71828...)
```

Référence

Log.

Sin (nombre) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Nombre, exprimé en radians, dont vous voulez connaître le sinus
Résultat	Numérique	←	Sinus de nombre

Description

Sin retourne le sinus de nombre. La valeur nombre est exprimée en radians.

Note : 4D fournit les constantes prédéfinies Pi, Degré et Radian. Pi retourne le nombre Pi (3,14159...), Degré retourne la valeur en radians d'un degré (0,01745...) et Radian retourne la valeur en degrés d'un radian (57,29577...).

Référence

Arctan, Cos, Tan.

Cos (nombre) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Nombre, exprimé en radians, dont vous voulez connaître le cosinus
Résultat	Numérique	←	Cosinus de nombre

Description

Cos retourne le cosinus de nombre. La valeur nombre est exprimée en radians.

Note : 4D fournit les constantes prédéfinies Pi, Degré et Radian. Pi retourne le nombre Pi (3,14159...), Degré retourne la valeur en radians d'un degré (0,01745...) et Radian retourne la valeur en degrés d'un radian (57,29577...).

Référence

Arctan, Sin, Tan.

Tan (nombre) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Nombre, exprimé en radians, dont vous voulez connaître la tangente
Résultat	Numérique	←	Tangente de nombre

Description

Tan retourne la tangente de nombre. La valeur nombre est exprimée en radians.

Note : 4D fournit les constantes prédéfinies Pi, Degré et Radian. Pi retourne le nombre Pi (3,14159...), Degré retourne la valeur en radians d'un degré (0,01745...) et Radian retourne la valeur en degrés d'un radian (57,29577...).

Référence

Arctan, Cos, Sin.

Arctan (nombre) → Numérique

Paramètre	Type		Description
nombre	Numérique	→	Tangente pour laquelle vous souhaitez calculer l'angle en radians
Résultat	Numérique	←	Angle en radians

Description

Arctan retourne en radians la valeur de l'angle dont la tangente est spécifiée par nombre.

Note : 4D fournit les constantes prédéfinies Pi, Degré et Radian. Pi retourne le nombre Pi (3,14159...), Degré retourne la valeur en radians d'un degré (0,01745...) et Radian retourne la valeur en degrés d'un radian (57,29577...).

Exemple

Cet exemple permet d'afficher la valeur de Pi :

```
⇒ ALERTE ("Pi est égal à : "+Chaine(Arctan(1)*4))
```

Référence

Cos, Sin, Tan.

FIXER NIVEAU COMPARAISON REEL (epsilon)

Paramètre	Type		Description
epsilon	Numérique	→	Valeur epsilon pour les comparaisons d'égalité des réels

Description

La commande FIXER NIVEAU COMPARAISON REEL définit la valeur epsilon utilisée par 4e Dimension lors d'une comparaison d'égalité des valeurs et expressions de type Réel. Comme un ordinateur effectue des calculs approximatifs sur les réels, les tests sur l'égalité de valeurs réelles doivent tenir compte de cette approximation. Pour cela, 4e Dimension, lorsqu'il compare des valeurs réelles, teste en fait si la différence entre les deux valeurs est supérieure ou non à une certaine valeur. Cette valeur s'appelle l'**epsilon** et fonctionne de la manière suivante : soient deux valeurs réelles a et b. Si Abs(a-b) est supérieur à l'epsilon, les valeurs sont considérées comme différentes ; sinon, elles sont déclarées égales.
Par défaut, 4e Dimension fixe la valeur epsilon à 10 à la puissance moins 6 (10^-6).
Exemples :

- 0,00001=0,00002 retourne Faux car la différence 0,00001 est supérieure à 10^-6.
- 0,000001=0,000002 retourne Vrai car la différence 0,000001 n'est pas supérieure à 10^-6.
- 0,000001=0,000003 retourne Faux car la différence 0,000002 est supérieure à 10^-6.

A l'aide de FIXER NIVEAU COMPARAISON REEL, vous pouvez augmenter ou réduire la valeur epsilon, en fonction de vos besoins.

Note : La commande n'aura pas d'effet si epsilon > 10^-3 ou si epsilon < 0.

Modifier l'epsilon affecte seulement la comparaison d'égalité des réels. Cela n'a pas d'effet sur les calculs et l'affichage des valeurs réelles.

ATTENTION : Cette commande doit être utilisée dans des cas spécifiques, comme par exemple un tri sur un champ dont les valeurs sont inférieures à 10^-6. En général, vous n'avez pas besoin de modifier la valeur par défaut d'epsilon.

Note : Si vous souhaitez effectuer un traitement (tri, recherche) sur un champ numérique indexé dont les valeurs sont inférieures à 10^-6, veillez à ce que la commande FIXER NIVEAU COMPARAISON REEL soit exécutée **avant** la construction de l'index.

Référence

Opérateurs de comparaison.

Note préliminaire

Si vous ne faites pas de développement multi-plate-forme, vous pouvez tout simplement ignorer cette section.

Dans le monde de l'informatique, l'arithmétique des nombres décimaux tient plus de la technologie que de la science exacte. Vous avez avez appris en classe, par exemple, qu'un tiers peut être écrit comme une suite infinie de 3 après la virgule. Un ordinateur, quant à lui, ne le sait pas mais doit cependant évaluer cette expression. De la même manière, vous savez que trois tiers sont égal à un ; l'ordinateur calculera l'expression pour obtenir le résultat. Suivant le type d'ordinateur que vous utilisez, un tiers sera évalué comme un nombre fini de trois après la virgule. Ce nombre est appelé la précision de la machine.

Sur les Macintosh à base de processeur 68K, ce nombre est égal à 19 ; cela signifie qu'un tiers sera évalué avec 19 chiffres significatifs. Sous Windows ou sur une machine à base de PowerPC, ce chiffre est égal à 15 ; cela signifie qu'un tiers sera évalué avec 15 chiffres significatifs. Si vous affichez la valeur 1/3 dans le débogueur de 4e Dimension, vous obtiendrez 0,3333333333333333333 sur les Macintosh à base de processeur 68K et environ 0,3333333333333333148 sous Windows ou sur une machine à base de processeur PowerPC. Il est à noter que les trois derniers chiffres sont différents car la précision des Macintosh à base de 68K est supérieure à celle des PC ou des machines à base de PowerPC. Cependant, si vous affichez l'expression (1/3)*3 vous obtenez 1 sur tous ces types de machines.

Si vos calculs décimaux se rapportent au nombre de mètres carrés de votre jardin, il est probable que vous n'aurez qu'un besoin très relatif des chiffres après la virgule. D'un autre côté, si vous remplissez une déclaration d'impôts, vous pouvez à certains moments vous sentir plus concerné par la précision de votre ordinateur. Cependant, une précision de 15 ou 19 est suffisante, même pour gérer des milliards de francs de revenus !

Pourquoi la valeur 1/3 semble-t-elle différente entre des Macintosh à base de processeur 68K et des machines Windows ou à base de PowerPC ?

Sur les Macintosh à base de processeur 68K, le système d'exploitation stocke les nombres réels sur 10 octets (80 bits), alors que sur un PC ou un Power Macintosh les nombres réels sont stockés sur 8 octets (64 bits). C'est pour cette raison que les nombres réels ont jusqu'à 19 chiffres significatifs sur des Macintosh à base de processeur 68K et jusqu'à 15 chiffres significatifs sur PC et Power Macintosh.

Pourquoi l'expression (1/3)*3 retourne-t-elle 1 sur ces trois types de machines ?

Un ordinateur ne peut faire que des calculs approximatifs. Par conséquent, lorsqu'il compare ou calcule des nombres, il ne traite pas les nombres réels comme des éléments mathématiques mais comme des valeurs approximatives. Dans l'exemple évoqué plus haut, 0,3333... multiplié par 3 est égal à 0,9999..., cette valeur est tellement proche de 1 que la machine considère que le résultat est égal à 1. Pour plus d'informations sur ce point, reportez vous à la commande FIXER NIVEAU COMPARAISON REEL.

Une distinction doit être établie entre :
• La manière dont les nombres réels sont calculés et comparés,
• La manière dont les nombres réels sont affichés à l'écran (ou imprimés).

A l'origine, 4e Dimension manipulait les nombres réels à l'aide du type standard de 10 octets correspondant au système d'exploitation des Macintosh 68K. De ce fait, les valeurs réelles sauvegardées sur disque sont sauvegardées dans ce format. Afin de maintenir une compatibilité entre toutes les versions (Windows, Power Macintosh et Macintosh) de 4e Dimension, les fichiers de données utilisent toujours le même type de format de 10 octets. Comme les calculs internes sont réalisés sur 8 octets pour les versions PC et Power Macintosh, 4e Dimension convertit en interne les valeurs de 10 à 8 octets, ou inversement. De ce fait, si vous chargez un enregistrement sous Windows ou sur une machine à base de PowerPC, et que cet enregistrement contient des valeurs réelles qui ont été sauvegardées sur une machine à base de processeur 68K, il est possible de perdre de la précision (en passant de 19 à 15 chiffres significatifs). Si vous chargez un enregistrement sur un Macintosh à base de processeur 68K et que cet enregistrement contient des valeurs réelles créées ou stockées sur une machine Windows ou à base de processeur PowerPC, il n'y aura aucune perte de précision. De manière générale, si vous utilisez une base de données sur des machines à base de processeur 68K, PowerPC ou une machine Windows, comptez sur une précision de 15 et non 19 chiffres.

A l'aide de Customizer Plus, vous pouvez définir le nombre de chiffres qui doivent être ignorés pour l'affichage des nombres réels à l'écran pour ces trois types de machines. Les réglages par défaut sont 0 pour les machines à base de processeur 68K et 5 pour les autres.

Convertisseur Euro (valeur; deMonnaie; versMonnaie) → Numérique

Paramètre	Type		Description
valeur	Numérique	→	Valeur à convertir
deMonnaie	Alpha	→	Code ISO de la monnaie dans laquelle la valeur est exprimée
versMonnaie	Alpha	→	Code ISO de la monnaie dans laquelle la valeur doit être convertie
Résultat	Numérique	←	Valeur convertie

Description

La commande Convertisseur Euro vous permet d'effectuer tout type de conversion de valeurs entre les différentes monnaies des pays de la "zone euro" et l'Euro lui-même.

Vous pouvez convertir :
• une monnaie nationale en Euros,
• des Euros en une monnaie nationale,
• une monnaie nationale en une autre monnaie nationale. Dans ce cas, la conversion s'effectue toujours par l'intermédiaire de l'Euro, comme le stipule la réglementation. Par exemple, pour convertir des Francs belges en Marks allemands, 4D effectuera les conversions suivantes : Francs belges -> Euro -> Marks allemands.

Vous passez dans le premier paramètre la valeur à convertir.
Dans le second paramètre, vous indiquez le code ISO de la monnaie dans laquelle valeur est exprimée.
Dans le troisième paramètre, vous indiquez le code ISO de la monnaie dans laquelle vous souhaitez que valeur soit convertie.

Pour désigner les codes ISO, 4e Dimension vous propose les constantes prédéfinies suivantes, placées dans le thème “Euro monnaies” :

Constante	Type	Valeur
Escudo portugais	Alpha	PTE
Euro	Alpha	EUR
Florin néerlandais	Alpha	NLG
Franc belge	Alpha	BEF
Franc français	Alpha	FRF
Franc luxembourgeois	Alpha	LUF
Lire italienne	Alpha	ITL
Livre irlandaise	Alpha	IEP
Mark allemand	Alpha	DEM
Mark finlandais	Alpha	FIM
Peseta espagnole	Alpha	ESP
Schilling autrichien	Alpha	ATS

Si nécessaire, 4e Dimension arrondit automatiquement le résultat des conversions et conserve 2 décimales — à l’exception des conversions vers les Lires italiennes, Francs luxembourgeois, Francs belges et Pesetas espagnoles, pour lesquelles 4D conserve 0 décimale (le résultat est un nombre entier).

La parité des différentes monnaies vis-à-vis de l’Euro est fixe. Les taux de conversion, utilisés en interne par 4e Dimension, sont les suivants :

Monnaie	Valeur pour 1 Euro
Escudo portugais	200,482
Florin néerlandais	2,20371
Franc belge	40,3399
Franc français	6,55957
Franc luxembourgeois	40,3399
Lire italienne	1936,27
Livre irlandaise	0,787564
Mark allemand	1,95583
Mark finlandais	5,94573
Peseta espagnole	166,386
Schilling autrichien	13,7603

Exemple

Voici différents types de conversion pouvant être obtenus à l’aide de cette commande :

```
$valeur:=10000  `Valeur exprimée en francs français
   `Convertir la valeur en euros
⇒  $EnEuros:=Convertisseur Euro ($valeur;Franc français; Euro)
   `Convertir la valeur en lires italiennes
⇒  $EnLires:=Convertisseur Euro ($valeur;Franc français; Lire italienne)
```

21

Fonctions statistiques

Les fonctions de ce thème effectuent des calculs sur des séries de valeurs.

Les fonctions Moyenne, Max, Min, Somme, Somme des carres, Ecart type et Variance peuvent s'appliquer aux champs et aux sous-champs. Dans le cas d'un champ, elles s'appliquent à une sélection d'enregistrements. Dans le cas d'un sous-champ, elles s'appliquent à une sélection de sous-enregistrements de l'enregistrement courant. Notez que les fonctions Somme des carres, Ecart type et Variance ne peuvent être utilisées pour un champ que pendant l'impression.

Les fonctions ne travaillent que sur des valeurs numériques. Chacune de ces fonctions retourne une valeur numérique.

Utiliser un champ

Lorsque les fonctions Moyenne, Max, Min ou Somme sont utilisées sur un champ en-dehors d'une opération d'impression, il se peut qu'elles aient à charger chaque enregistrement de la sélection courante pour calculer le résultat. S'il y a beaucoup d'enregistrements, l'opération peut être longue. Pour éviter le problème, indexez le champ.

Lorsque ces fonctions sont utilisées dans un état, elles se comportent différemment, car c'est l'état lui-même qui charge chaque enregistrement. Utilisez ces fonctions dans une méthode formulaire ou objet quand vous imprimez avec la commande IMPRIMER SELECTION ou quand vous imprimez en mode Utilisation à l'aide de la commande **Imprimer** dans le menu **Fichier**.
Lorsque vous utilisez ces fonctions dans un état, les valeurs retournées sont fiables uniquement dans le niveau 0 de rupture et seulement si la phase de traitement des ruptures est active. Cela signifie qu'elles ne sont utiles qu'en fin d'état, lorsque tous les enregistrements ont été traités.

Vous ne devez utiliser ces fonctions avec une méthode objet que dans une zone non-saisissable incluse dans la zone de rupture R0.

Pour mémoire : un champ passé comme paramètre à une fonction statistique doit être un numérique.

Référence

Ecart type, Max, Min, Moyenne, Somme, Somme des carres, Variance.

Somme (séries) → Numérique

Paramètre	Type		Description
séries	Champ | Sous-chp	→	Champ ou sous-champ dont vous souhaitez calculer la somme
Résultat	Numérique	←	Somme de séries

Description

Somme retourne la somme (c'est-à-dire le total de toutes les valeurs) de **séries**. Si **séries** est un champ indexé, l'index est utilisé pour le calcul.

Exemples

L'exemple ci-dessous est la méthode objet d'une variable, vTotal, placée dans un formulaire. La méthode assigne à la variable la somme de tous les salaires :

```
⇒ vTotal := Somme ([Employés]Salaire)
```

La méthode suivante est appelée pour imprimer les enregistrements de la sélection courante et activer le traitement des ruptures :

```
TOUT SELECTIONNER ([Employés])
TRIER ([Employés];[Employés]Nom;>)
NIVEAUX DE RUPTURES (1)
CUMULER SUR ([Employés]Salaire)
FORMULAIRE SORTIE ([Employés];"FormImpression")
IMPRIMER SELECTION ([Employés])
```

Note : La valeur du paramètre de la commande NIVEAUX DE RUPTURES doit être égale au nombre de ruptures que contient l'état. Pour plus d'informations sur les ruptures, reportez-vous aux commandes du thème Impressions.

Référence

CUMULER SUR, IMPRIMER SELECTION, Max, Min, Moyenne, NIVEAUX DE RUPTURES, Sous total, TRIER.

Moyenne (séries) → Numérique

Paramètre	Type		Description
séries	Champ | Sous-chp	→	Champ ou sous-champ pour lequel vous voulez calculer la moyenne
Résultat	Numérique	←	Moyenne arithmétique de séries

Description

Moyenne retourne la moyenne arithmétique de séries. Si séries est un champ indexé, l'index est utilisé pour le calcul.

Exemple

Dans l'exemple suivant, une valeur est assignée à une variable se trouvant dans la zone de rupture R0 d'un formulaire sortie. La ligne de code ci-dessous constitue la méthode objet de la variable. Elle n'est exécutée qu'à l'impression du niveau de rupture 0 :

```
⇒ vMoyenne := Moyenne ([Employés]Salaire)
```

La méthode suivante est appelée pour imprimer les enregistrements de la sélection courante et activer la phase de rupture :

```
TOUT SELECTIONNER ([Employés])
TRIER ([Employés];[Employés]Nom;>)
NIVEAUX DE RUPTURES (1)
CUMULER SUR ([Employés]Salaire)
FORMULAIRE SORTIE ([Employés];"FormImpression")
IMPRIMER SELECTION ([Employés])
```

Note : La valeur du paramètre de la commande NIVEAUX DE RUPTURES doit être égale au nombre de ruptures que contient l'état. Pour plus d'informations sur les ruptures, reportez-vous aux commandes du thème Impressions.

Référence

CUMULER SUR, IMPRIMER SELECTION, Max, Min, NIVEAUX DE RUPTURES, Somme, Sous total, TRIER.

Min (séries) → Numérique

Paramètre	Type		Description
séries	Champ | Sous-chp	→	Champ ou sous-champ pour lequel vous voulez obtenir la valeur la plus basse
Résultat	Numérique	←	Valeur la plus basse de séries

Description

Min retourne la valeur la plus faible de séries. Si séries est un champ indexé, l'index est utilisé pour la recherche de cette valeur.

Exemples

(1) L'exemple ci-dessous est la méthode objet d'une variable, vMin, placée dans la zone de rupture R0 d'un formulaire sortie. La variable est imprimée à la fin de l'état. La méthode objet assigne la valeur la plus basse du champ à la variable, qui est alors imprimée dans la dernière rupture de l'état.

```
⇒ vMin := Min ([Employés] Salaire)
```

La méthode suivante est appelée pour imprimer les enregistrements de la sélection courante et activer la phase de rupture :

```
TOUT SELECTIONNER ([Employés])
TRIER ([Employés];[Employés]Nom;>)
NIVEAUX DE RUPTURES (1)
CUMULER SUR ([Employés]Salaire)
FORMULAIRE SORTIE ([Employés];"FormImpression")
IMPRIMER SELECTION ([Employés])
```

Note : La valeur du paramètre de la commande NIVEAUX DE RUPTURES doit être égale au nombre de ruptures que contient l'état. Pour plus d'informations sur les ruptures, reportez-vous aux commandes du thème Impressions.

(2) L'exemple suivant recherche la plus petite vente d'un employé et affiche le résultat dans une boîte de dialogue d'alerte. Les montants des ventes sont stockés dans un sous-champ, [Employés]VentesFrancs :

```
⇒ ALERTE ("Plus petite vente = " + Chaine(Min([Employés]VentesFrancs)))
```

Max (séries) → Numérique

Paramètre	Type		Description
séries	Champ | Sous-chp	→	Champ ou sous-champ dont vous voulez obtenir la valeur la plus élevée
Résultat	Numérique	←	Valeur la plus élevée de séries

Description

Max retourne la valeur la plus élevée contenue dans séries. Si séries est un champ indexé, l'index est utilisé pour la recherche de cette valeur.

Exemple

L'exemple ci-dessous est la méthode objet d'une variable, vMax, placée dans la zone de rupture R0 d'un formulaire sortie. La variable est imprimée à la fin de l'état. La méthode objet assigne la valeur la plus élevée du champ à la variable, qui est alors imprimée dans la dernière rupture de l'état.

```
⇒ vMax := Max ([Employés] Salaire)
```

La méthode suivante est appelée pour imprimer les enregistrements de la sélection courante et activer la phase de rupture :

```
TOUT SELECTIONNER ([Employés])
TRIER ([Employés];[Employés]Nom;>)
NIVEAUX DE RUPTURES (1)
CUMULER SUR ([Employés]Salaire)
FORMULAIRE SORTIE ([Employés];"FormImpression")
IMPRIMER SELECTION ([Employés])
```

Note : La valeur du paramètre de la commande NIVEAUX DE RUPTURES doit être égale au nombre de ruptures que contient l'état. Pour plus d'informations sur les ruptures, reportez-vous aux commandes du thème Impressions.

Référence

Min.

Ecart type (séries) → Numérique

Paramètre	Type		Description
séries	Champ | Sous-chp	→	Champ ou sous-champ dont vous voulez obtenir l'écart type
Résultat	Numérique	←	Ecart type de séries

Description

Ecart type retourne l'écart type (c.-à-d. la racine carrée de la variance) de séries. Si séries est un champ indexé, l'index sera utilisé pour le calcul. Lorsqu'elle est appelée lors de l'impression d'un état, cette fonction ne peut être appliquée qu'à un champ.

Exemple

L'exemple suivant est la méthode objet d'une variable appelée EcartT. La méthode assigne l'écart type d'une série à EcartT :

```
⇒ EcartT := Ecart type ([Table1]SérieValeurs)
```

La méthode suivante est appelée pour imprimer les enregistrements de la sélection et activer la phase de rupture :

```
TOUT SELECTIONNER ([Table1])
TRIER ([Table1];[Table1]SérieValeurs;>)
NIVEAUX DE RUPTURES (1)
CUMULER SUR ([Table1]SérieValeurs)
FORMULAIRE SORTIE ([Table1];"FormImpression")
IMPRIMER SELECTION ([Table1])
```

Note : La valeur du paramètre de la commande NIVEAUX DE RUPTURES doit être égale au nombre de ruptures que contient l'état. Pour plus d'informations sur les ruptures, reportez-vous aux commandes du thème Impressions.

Référence

Moyenne, Somme, Somme des carres, Variance.

Variance (séries) → Numérique

Paramètre	Type		Description
séries	Champ | Sous-chp	→	Champ ou sous-champ dont vous voulez obtenir la variance
Résultat	Numérique	←	Variance de séries

Description

Variance retourne la variance de séries. Si séries est un champ indexé, l'index est utilisé pour le calcul. Lorsqu'elle est appelée lors de l'impression d'un état, cette fonction ne peut être appliquée qu'à un champ.
La variance d'un ensemble de valeurs est la moyenne des carrés des écarts par rapport à la moyenne. C'est une valeur de dispersion autour de la moyenne. 4D utilise la formule de variance suivante :
Variance(x) = Somme (x-m)*(x-m)/(n-1)
m = Moyenne
n = Nombre de valeurs
Si les valeurs considérées ne constituent pas un échantillon, multipliez la valeur retournée par Variance par (n-1)/n.

Exemple

L'exemple suivant est la méthode objet d'une variable appelée Var. La méthode assigne la variance de la série à Var:

```
⇒ Var:= Variance ([Etudiants]Notes)
```

La méthode suivante est appelée pour imprimer les enregistrements de la sélection et activer la phase de rupture :

```
TOUT SELECTIONNER ([Etudiants])
TRIER ([Etudiants];[Etudiants]Classe;>)
NIVEAUX DE RUPTURES (1)
CUMULER SUR ([Etudiants]Notes)
FORMULAIRE SORTIE ([Etudiants];"FormImpression")
IMPRIMER SELECTION ([Etudiants])
```

Note : La valeur du paramètre de la commande NIVEAUX DE RUPTURES doit être égale au nombre de ruptures que contient l'état. Pour plus d'informations sur les ruptures, reportez-vous aux commandes du thème Impressions.

Référence

Ecart type, Moyenne, Somme, Somme des carres.

Somme des carres (séries) → Numérique

Paramètre	Type		Description
séries	Champ | Sous-chp	→	Champ ou sous-champ dont vous voulez obtenir la somme des carrés
Résultat	Numérique	←	Somme des carrés de séries

Description

Somme des carres retourne la somme des carrés de séries. Si séries est un champ indexé, l'index est utilisé pour le calcul. Lorsqu'elle est appelée lors de l'impression d'un état, cette fonction ne peut être appliquée qu'à un champ.

Exemple

L'exemple suivant est la méthode objet d'une variable appelée Carrés. La méthode assigne la somme des carrés d'une série de valeurs à Carrés. La méthode est imprimée dans la dernière rupture de l'état :

```
⇒    Carrés := Somme des carres ([Table1]SérieValeurs)
```

La méthode suivante est appelée pour imprimer les enregistrements de la sélection et activer la phase de rupture :

```
TOUT SELECTIONNER ([Table1])
TRIER ([Table1];[Table1]SérieValeurs;>)
NIVEAUX DE RUPTURES (1)
CUMULER SUR ([Table1]SérieValeurs)
FORMULAIRE SORTIE ([Table1];"FormImpression")
IMPRIMER SELECTION ([Table1])
```

Note : La valeur du paramètre de la commande NIVEAUX DE RUPTURES doit être égale au nombre de ruptures que contient l'état. Pour plus d'informations sur les ruptures, reportez-vous aux commandes du thème Impressions.

Référence

Ecart type, Moyenne, Somme, Variance.

22

Gestion de la saisie

VALIDER

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

La commande VALIDER doit être employée dans une méthode objet ou formulaire (ou une sous-routine) pour :

• valider un enregistrement ou un sous-enregistrement créé ou modifié — dont les données ont été saisies à la suite d'un AJOUTER ENREGISTREMENT, MODIFIER ENREGISTREMENT, AJOUTER SOUS ENREGISTREMENT ou MODIFIER SOUS ENREGISTREMENT.
• valider un formulaire affiché par l'intermédiaire de la commande DIALOGUE.
• quitter un formulaire affichant une sélection d'enregistrements — à l'aide de VISUALISER SELECTION ou MODIFIER SELECTION.

VALIDER effectue la même action que lorsque l'utilisateur appuie sur la touche **Entrée**. Une fois que le formulaire a été validé, la variable système OK prend la valeur 1.

VALIDER est fréquemment exécutée à la suite de la sélection d'une commande de menu. VALIDER est également souvent appelée dans la méthode objet d'un bouton auquel la propriété "Pas d'action" a été associée.

Enfin, cette commande peut être placée dans la méthode de la case de fermeture (optionnelle) d'une fenêtre créée par la commande Creer fenetre. Si la fenêtre comporte une case de menu Système, VALIDER et NE PAS VALIDER peuvent être appelées dans la méthode à exécuter lorsque l'utilisateur double-clique sur la case du menu Système ou sélectionne la commande de menu **Fermeture**.

Il n'est pas possible d'enchaîner plusieurs VALIDER. En d'autres termes, l'exécution consécutive de deux commandes VALIDER dans une méthode en réponse à un événement aura le même résultat que l'exécution d'une seule.

Référence

NE PAS VALIDER.

NE PAS VALIDER

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

La commande NE PAS VALIDER doit être employée dans une méthode objet ou formulaire (ou une sous-routine) pour :

• annuler la création ou la modification d'un enregistrement ou un sous-enregistrement — dont les données ont été saisies à la suite d'un AJOUTER ENREGISTREMENT, MODIFIER ENREGISTREMENT, AJOUTER SOUS ENREGISTREMENT ou MODIFIER SOUS ENREGISTREMENT.
• annuler un formulaire affiché par l'intermédiaire de la commande DIALOGUE.
• quitter un formulaire affichant une sélection d'enregistrements — à l'aide de VISUALISER SELECTION ou MODIFIER SELECTION.

NE PAS VALIDER effectue la même action que lorsque l'utilisateur utilise la touche d'annulation (**Esc**). Une fois que le formulaire a été annulé, la variable système OK prend la valeur 0.

NE PAS VALIDER est fréquemment exécutée à la suite de la sélection d'une commande de menu. NE PAS VALIDER est également souvent appelée dans la méthode objet d'un bouton auquel la propriété "Pas d'action" a été associée.

Enfin, cette commande peut être placée dans la méthode de la case de fermeture (optionnelle) d'une fenêtre créée par la commande Creer fenetre. Si la fenêtre comporte une case de menu Système, NE PAS VALIDER et VALIDER peuvent être appelées dans la méthode à exécuter lorsque l'utilisateur double-clique sur la case du menu Système ou sélectionne la commande de menu **Fermeture**.

Il n'est pas possible d'enchaîner plusieurs NE PAS VALIDER. En d'autres termes, l'exécution consécutive de deux commandes NE PAS VALIDER dans une méthode en réponse à un événement aura le même résultat que l'exécution d'une seule.

Référence

VALIDER.

Frappe clavier → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Caractère saisi par l'utilisateur

Description

Frappe clavier retourne le caractère tapé par l'utilisateur dans un champ ou une zone saisissable.

En général, vous appelez Frappe clavier dans une méthode formulaire ou objet, lors de la gestion des événements formulaire Sur avant frappe clavier et Sur après frappe clavier. Pour détecter les événements de frappe clavier, utilisez la commande Evenement formulaire.

Si vous voulez remplacer un caractère saisi par l'utilisateur par un autre, utilisez la commande FILTRER FRAPPE CLAVIER.

IMPORTANT : Si vous voulez effectuer des opérations “à la volée” en fonction de la valeur courante de la zone saisissable en cours de modification ainsi que du caractère à saisir, rappelez-vous que le texte affiché à l'écran n'est pas encore la valeur du champ ou de la variable. La valeur saisie dans une variable ou un champ ne lui est affectée que lorsque la zone est validée (si l'utilisateur appuie sur la touche Tabulation, clique sur un bouton, etc.). En conséquence, pensez à placer les valeurs saisies dans une variable temporaire et à travailler avec celle-ci, ou utilisez la commande Lire texte edite. Vous devez procéder ainsi si vous souhaitez connaître la valeur courante du texte pour effectuer des actions spéciales.

Vous pouvez utiliser la commande Frappe clavier pour :
• effectuer un filtrage personnalisé des caractères
• créer un filtre de saisie non disponible en standard, par exemple dans les filtres de saisie
• implémenter des zones de recherche ou de pré-saisie dynamiques.

Note : Vous ne pouvez pas utiliser la fonction Frappe clavier dans les sous-formulaires.

Exemples

(1) Référez-vous aux exemples de la commande FILTRER FRAPPE CLAVIER.

(2) Lorsque vous traitez un événement Sur avant frappe clavier, vous gérez la modification de la zone de texte courante (celle qui contient le curseur), et non la "valeur future" de la source de données (champ ou variable) de cette zone. La méthode Gérer frappe clavier décrite ci-dessous vous permet de placer dans une seconde variable les caractères saisis dans une zone de texte. Vous pouvez alors utiliser cette variable pour effectuer différentes actions pendant la saisie des caractères dans la zone. Vous passez comme premier paramètre un pointeur vers la source des données de la zone, et comme second paramètre un pointeur vers cette seconde variable. La méthode renvoie la nouvelle valeur de la zone de texte dans la seconde variable et retourne Vrai si cette valeur est différente de ce qu'elle était avant la saisie du dernier caractère.

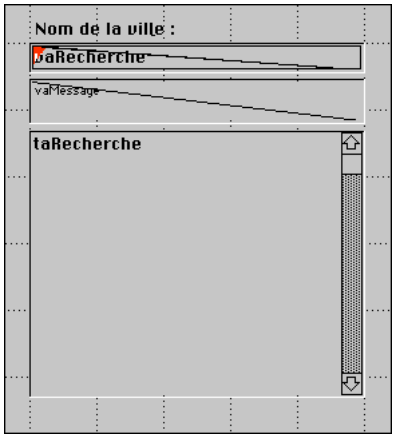
```
      ` Méthode projet Gérer frappe clavier
      ` Gérer frappe clavier ( Pointeur ; Pointeur ) -> Booléen
      ` Gérer frappe clavier ( -> zoneSource ; -> valeurCourante ) -> Nouvelle valeur ?

   C_POINTEUR ($1;$2)
   C_TEXTE ($vtNouvValeur)

      ` Récupérer le texte sélectionné dans la zone saisissable
   TEXTE SELECTIONNE ($1->;$vlDébut;$vlFin)
      ` Commencer à travailler avec la valeur courante
   $vtNouvValeur:=$2->
      ` Selon la touche appuyée ou le caractère saisi, effectuer les actions appropriées
   Au cas ou

         ` La touche Retour arrière a été enfoncée
⇒       : (Code ascii (Frappe clavier)=Touche Retour arrière )
            ` Supprimer les caractères sélectionnés ou le caractère à gauche du curseur
          $vtNouvValeur:=Sous chaine ($vtNouvValeur;1;$vlDébut-1-Num($vlDébut=
                                                $vlFin))+Sous chaine($vtNouvValeur;$vlFin)

         ` Un caractère acceptable a été saisi
⇒       : (Position (Frappe clavier;"abcdefghjiklmnopqrstuvwxyz -0123456789")>0)
          Si ($vlDébut#$vlFin)
               ` Un ou plusieurs caractères sont sélectionnés, la frappe clavier va les effacer
⇒           $vtNouvValeur:=Sous chaine($vtNouvValeur;1;$vlDébut-1)
                           +Frappe clavier+Sous chaine($vtNouvValeur;$vlFin)
          Sinon
               ` La sélection de texte est le curseur
            Au cas ou
                  ` Le curseur est actuellement au début du texte
              : ($vlDébut<=1)
                     ` Insertion du caractère au début du texte
⇒               $vtNouvValeur:=Frappe clavier+$vtNouvValeur
                     ` Le curseur est actuellement à la fin du texte
              : ($vlDébut>=Longueur($vtNouvValeur))
```

```
                        ` Ajouter le caractère à la fin du texte
⇒                   $vtNouvValeur:=$vtNouvValeur+Frappe clavier
                Sinon
                        ` Le curseur se trouve dans le texte, insérer le nouveau caractère
⇒                   $vtNouvValeur:=Sous chaine($vtNouvValeur;1;$vlDébut-1)+
                                        Frappe clavier+Sous chaine($vtNouvValeur;$vlDébut)
                Fin de cas
            Fin de si

            ` Une touche flèche a été enfoncée
            ` Ne rien faire, mais valider la frappe clavier
⇒       : (Code ascii(Frappe clavier)=Touche gauche )
⇒       : (Code ascii(Frappe clavier)=Touche droite )
⇒       : (Code ascii(Frappe clavier)=Touche haut )
⇒       : (Code ascii(Frappe clavier)=Touche bas )
        `
    Sinon
        ` Ne pas accepter des caractères autres que des lettres, chiffres, espaces et tirets
        FILTRER FRAPPE CLAVIER ("")
    Fin de cas
        ` Est-ce que la valeur est maintenant différente ?
    $0:=($vtNouvValeur#$2->)
        ` Retourner la valeur pour la gestion de la prochaine frappe clavier
    $2->:=$vtNouvValeur
```

Une fois que vous avez ajouté cette méthode projet à votre application, vous pouvez l'utiliser ainsi :

```
        ` Méthode objet de la zone saisissable MonObjet
    Au cas ou
        : (Evenement formulaire=Sur chargement)
            MonObjet:=""
            MonObjetCaché:=""
        : (Evenement formulaire=Sur avant frappe clavier)
            Si (Gérer frappe clavier (->MonObjet;->MonObjetCaché))
                ` Effectuer des actions appropriées par rapport
                ` à la valeur stockée dans MonObjetCaché
            Fin de si
    Fin de cas
```

Examinons par exemple le formulaire suivant :

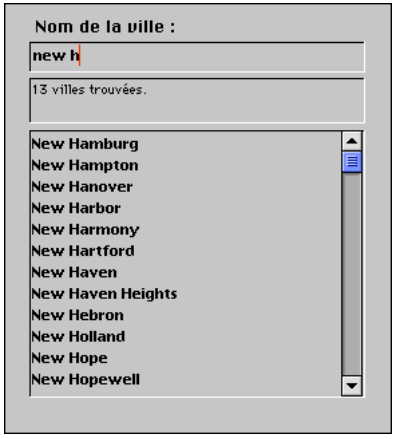


Il est composé des objets suivants : une zone saisissable vaRecherche, une zone non-saisissable vaMessage et une zone de défilement taRecherche. Lorsque l'utilisateur saisit des caractères dans vaRecherche, la méthode objet effectue une recherche sur la table [Codes postaux] permettant d'afficher des villes américaines en saisissant seulement les premiers caractères de leur nom. Voici la méthode objet de vaRecherche :

```
   ` Méthode objet de la zone saisissable vaRecherche
Au cas ou
   : (Evenement formulaire=Sur chargement )
      vaRecherche:=""
      vaRésultat:=""
      vaMessage:="Saisissez les premiers caractères de la ville que vous cherchez."
      EFFACER VARIABLE(taRecherche)
   : (Evenement formulaire=Sur avant frappe clavier )
      Si (Gérer frappe clavier (->vaRecherche;->vaRésultat))
         Si (vaRésultat#"")
            CHERCHER([Codes postaux];[Codes postaux]Ville=vaRésultat+"@")
            SUPPRIMER MESSAGES
            VALEURS DISTINCTES([Codes postaux]Ville;taRecherche)
            LAISSER MESSAGES
            $vlRésultat:=Taille tableau(taRecherche)
            Au cas ou
               : ($vlRésultat=0)
                  vaMessage:="Aucune ville trouvée."
               : ($vlRésultat=1)
                  vaMessage:="Une ville trouvée."
            Sinon
               vaMessage:=Chaine($vlRésultat)+" villes trouvées."
            Fin de cas
```

```
            Sinon
               SUPPRIMER LIGNES(taRecherche;1;Taille tableau(taRecherche))
               vaMessage:="Saisissez les premières lettres de la ville que vous cherchez."
            Fin de si
         Fin de si
      Fin de cas
```

Voici le formulaire en mode Utilisation :



A l'aide des possibilités de communication interprocess de 4e Dimension, vous pouvez construire une interface dans laquelle les recherches se construisent dans des palettes flottantes communiquant avec les process dans lesquels les enregistrements sont affichés ou modifiés.

Référence

Evenement formulaire, FILTRER FRAPPE CLAVIER, Lire texte edite.

FILTRER FRAPPE CLAVIER (carFiltré)

Paramètre	Type		Description
carFiltré	Alpha	→	Caractère(s) de remplacement ou Chaîne vide pour annuler le filtrage clavier

Description

FILTRER FRAPPE CLAVIER vous permet de remplacer le caractère saisi par l'utilisateur dans un champ ou une zone saisissable par le premier caractère de la chaîne carFiltré.

Si vous passez une chaîne vide, le filtrage clavier en cours est annulé.

Vous appelez généralement FILTRER FRAPPE CLAVIER dans une méthode formulaire ou objet lorsque vous gérez l'événement formulaire Sur avant frappe clavier. Pour détecter les événements de frappe clavier, utilisez la commande Evenement formulaire. Pour récupérer les caractères saisis au clavier, utilisez la fonction Frappe clavier.

IMPORTANT : Si vous voulez effectuer des opérations “à la volée” en fonction de la valeur courante de la zone saisissable en cours de modification ainsi que du caractère à saisir, rappelez-vous que le texte affiché à l'écran n'est pas encore la valeur du champ ou de la variable. La valeur saisie dans une variable ou un champ ne lui est affectée que lorsque la zone est validée (lorsque l'utilisateur appuie sur la touche **Tabulation**, clique sur un bouton, etc.). En conséquence, pensez à placer les valeurs saisies dans une variable temporaire et à travailler avec celle-ci, puis à assigner cette variable à la zone de saisie (reportez-vous à l'exemple ci-dessous). Vous pouvez également utiliser la fonction Lire texte edite.

Utilisez la commande FILTRER FRAPPE CLAVIER dans les cas suivants :
• Pour effectuer un filtrage personnalisé des caractères,
• Pour créer un filtre de saisie non disponible en standard,
• Pour implémenter des zones de recherche ou de pré-saisie dynamiques.

ATTENTION : si vous appelez la commande Frappe clavier après avoir appelé FILTRER FRAPPE CLAVIER, c'est le caractère passé à cette commande sera retourné et non le caractère réellement saisi.

Exemples

(1) Avec le code suivant :

```
      ` Méthode objet de la zone saisissable monObjet
   Au cas ou
      : (Evenement formulaire=Sur chargement )
      monObjet:=""
      : (Evenement formulaire=Sur avant frappe clavier )
         Si(Position(Frappe clavier;"0123456789")>0)
⇒           FILTRER FRAPPE CLAVIER("*")
         Fin de si
   Fin de cas
```

... tous les chiffres saisis dans la zone monObjet seront transformés en astérisques.

(2) Le code ci-dessous définit le comportement d'une zone de saisie de mot de passe, dans laquelle les caractères saisis sont remplacés à l'écran par des caractères aléatoires :

```
      ` Méthode objet de la zone saisissable vaMotsPasse
   Au cas ou
      : (Evenement formulaire=Sur chargement)
         vaMotsPasse:=""
         vaMotPasseActuel:=""
      : (Evenement formulaire=Sur avant frappe clavier)
         Gérer frappe clavier (->vaMotsPasse;->vaMotPasseRéel)
         Si (Position(Frappe clavier;Caractere(Touche Retour arrière)+
                                     Caractere(Touche gauche )+Caractere(Touche droite )+
                                       Caractere(Touche haut )+Caractere(Touche bas ))=0)
⇒           FILTRER FRAPPE CLAVIER(Caractere(65+(Hasard%26)))
         Fin de si
   Fin de cas
```

Une fois la zone validée, vous récupérez le mot de passe réellement saisi par l'utilisateur dans la variable vaMotPasseRéel. La méthode Gérer frappe clavier est listée dans l'exemple de la commande Frappe clavier.

(3) Vous disposez dans votre application de diverses zones de texte dans lesquelles vous pouvez saisir quelques phrases. Votre application comporte également une table de glossaire contenant les termes les plus fréquemment utilisés dans votre base. Lors de l'édition de vos zones de texte, vous voulez pouvoir rapidement, à partir du glossaire, retrouver et insérer des mots en fonction des caractères sélectionnés dans le texte. Pour cela, vous avez deux solutions : soit placer des boutons avec des touches associées qui vont exécuter l'opération, soit intercepter les frappes clavier spéciales pendant la saisie. L'exemple ci-dessous utilise la seconde solution, basée sur la touche Aide.

Comme décrit ci-dessus, lorsque vous éditez une zone de texte, la valeur du champ ou de la variable de texte ne sera réellement modifiée que lorsque que vous l'aurez validée. Pour retrouver et insérer rapidement des entrées du glossaire dans une zone de texte alors qu'elle est en train d'être modifiée, vous devez donc créer une seconde zone "tampon" pour y placer les valeurs saisies. Vous pouvez effectuer cette opération à l'aide de la méthode projet décrite ci-dessous. Vous passez comme premiers paramètres des pointeurs vers la zone de saisie et vers la variable, puis la chaîne de caractère “interdits” comme troisième paramètre. Peu importe comment l'entrée clavier sera traitée, la méthode retourne la valeur saisie originale. Les caractères “interdits” sont les caractères que vous ne voulez pas insérer dans la zone saisissable et que vous voulez traiter en tant que caractères spéciaux.

```
   ` Méthode projet Frappe clavier tampon
   ` Frappe clavier tampon ( Pointeur ; Pointeur ; Alpha ) -> Alpha
   ` Frappe clavier tampon ( -> zoneSource ; -> valeurCourante ; Filtre )
                                                          -> Ancien frappe clavier
C_ALPHA(1;$0)
C_POINTEUR($1;$2)
C_TEXTE($vtNouvValeur)
C_ALPHA(255;$3)
   ` Retourne la frappe clavier originale
$0:=Frappe clavier
   ` Obtenir la sélection de texte dans la zone saisissable
TEXTE SELECTIONNE($1->;$vlDébut;$vlFin)
   ` Commencer à travailler sur la valeur courante
$vtNouvValeur:=$2->
   ` En fonction de la touche ou du caractère tapé(e), effectuer les actions appropriées
Au cas ou
      ` La touche Retour arrière a été enfoncée
   : (Code ascii($0)=Touche Retour arrière )
         ` Supprimer les caractères sélectionnés ou le caractère à gauche du curseur
      $vtNouvValeur:=Supprimer texte ($vtNouvValeur;$vlDébut;$vlFin)
         ` Une touche "flèche" a été appuyée
         ` Ne faites rien sauf accepter la frappe clavier
   : (Code ascii($0)=Touche gauche )
   : (Code ascii($0)=Touche droite )
   : (Code ascii($0)=Touche haut )
   : (Code ascii($0)=Touche bas )
```

```
            ` Un caractère valide a été saisi
        : (Position($0;$3)=0)
            $vtNouvValeur:=Inserer texte ($vtNouvValeur;$vlDébut;$vlFin;$0)
    Sinon
            ` Le caractère n'est pas accepté
⇒       FILTRER FRAPPE CLAVIER("")
    Fin de cas
        ` Retourner la valeur pour la prochaine gestion de la frappe clavier
    $2->:=$vtNouvValeur
```

Cette méthode utilise les sous-méthodes suivantes :

```
        ` Méthode projet Supprimer texte
        ` Supprimer texte ( Alpha ; Long ; Long ) -> Alpha
        ` Supprimer texte ( -> Texte ; SelDébut ; SelFin ) -> Nouveau texte
    C_TEXTE($0;$1)
    C_ENTIER LONG($2;$3)
    $0:=Sous chaine($1;1;$2-1-Num($2=$3))+Sous chaine($1;$3)
```

```
        ` Méthode projet Inserer texte
        ` Inserer texte ( Alpha ; Long ; Long ; Alpha ) -> Alpha
        ` Inserer texte ( -> texteSource ; SelDébut ; SelFin ; Texte à insérer ) -> Nouveau texte
    C_TEXTE($0;$1;$4)
    C_ENTIER LONG($2;$3)
    $0:=$1
    Si ($2#$3)
        $0:=Sous chaine($0;1;$2-1)+$4+Sous chaine($0;$3)
    Sinon
        Au cas ou
            : ($2<=1)
                $0:=$4+$0
            : ($2>Longueur($0))
                $0:=$0+$4
        Sinon
            $0:=Sous chaine($0;1;$2-1)+$4+Sous chaine($0;$2)
        Fin de cas
    Fin de si
```

Une fois que vous avez ajouté ces méthodes projet à votre base, vous pouvez les utiliser de la manière suivante :

```
   ` Méthode objet de la zone saisissable vaDescription
Au cas ou
   : (Evenement formulaire=Sur chargement )
      vaDescription:=""
      vaDescriptionDouble:=""
         ` Etablir la liste des caractères “interdits” à traiter comme des touches
         ` spéciales (dans cet exemple, seule la touche Aide est filtrée )
      vaTouchesSpéciales:=Caractere(Touche Aide)
   : (Evenement formulaire=Sur avant frappe clavier )
      $vsKey:=Frappe clavier tampon (->vaDescription;->vaDescriptionDouble;
                                                          vaTouchesSpéciales)
      Au cas ou
         : (Code ascii($vsKey)=Touche Aide)
            ` Faire quelque chose lorsque la touche Aide est enfoncée
            ` Dans cet exemple, une saisie de glossaire doit être recherchée et insérée
            chercher_Glossaire (->vaDescription;->vaDescriptionDouble)
      Fin de cas
Fin de cas
```

La méthode projet chercher_Glossaire est listée ci-dessous (le point principal est l'utilisation de la variable tampon pour réaffecter la zone saisissable à modifier) :

```
   ` Méthode projet chercher_Glossaire
   ` chercher_Glossaire ( Pointeur ; Pointeur )
   ` chercher_Glossaire ( -> zone saisissable ; ->variable double )

C_POINTEUR($1;$2)
C_ENTIER LONG($vlDébut;$vlFin)

   ` Obtenir la sélection de texte dans la zone saisissable
TEXTE SELECTIONNE($1->;$vlDébut;$vlFin)
   ` Obtenir le texte sélectionné ou le mot situé à gauche du curseur
$vtTexteSelectionne:=obtenirTexteSelectionne ($2->;$vlDébut;$vlFin)
   ` Y a-t-il quelque chose à rechercher ?
Si ($vtTexteSelectionne#"")
      ` Si la sélection de texte était le curseur, la sélection débute au mot situé après
      ` le curseur
   Si ($vlDébut=$vlFin)
      $vlDébut:=$vlDébut-Longueur($vtTexteSelectionne)
   Fin de si
      ` Chercher la première entrée du glossaire disponible
   CHERCHER([Glossaire];[Glossaire]Saisie=$vtTexteSelectionne+"@")
```

```
            ` Existe-t-elle ?
        Si (Enregistrements trouves([Glossaire])>0)
            ` Si oui, l'insérer dans la zone tampon
        $2->:=Inserer texte ($2->;$vlDébut;$vlFin;[Glossaire]Saisie)
            ` Copier le tampon dans la zone saisissable
        $1->:=$2->
            ` Fixer la sélection après avoir inséré l'entrée du glossaire
        $vlFin:=$vlDébut+Longueur([Dictionnaire]Saisie)
        SELECTIONNER TEXTE(vsComments;$vlFin;$vlFin)
    Sinon
            ` Il n'y a pas d'entrée qui correspond dans le glossaire
        BEEP
    Fin de si
Sinon
        ` Il n'y a pas de texte sélectionné
    BEEP
Fin de si
```

La méthode obtenirTexteSelectionne est la suivante :

```
    ` Méthode objet obtenirTexteSelectionne
    ` obtenirTexteSelectionne ( Alpha ; Entier long ; Entier long ) -> Alpha
    ` obtenirTexteSelectionne ( Texte ; SelDébut ; SelFin ) -> texte sélectionné
C_TEXTE($0;$1)
C_ENTIER LONG($2;$3)
Si ($2<$3)
    $0:=Sous chaine($1;$2;$3-$2)
Sinon
    $0:=""
    $2:=$2-1
    Repeter
        Si ($2>0)
            Si (Position($1[[$2]];"  ,.!?:;()-_——")=0)
                $0:=$1[[$2]]+$0
                $2:=$2-1
            Sinon
                $2:=0
            Fin de si
        Fin de si
    Jusque ($2=0)
Fin de si
```

Référence

Evenement formulaire, Frappe clavier, Lire texte edite.

ALLER A CHAMP ({*; }objet)

Paramètre	Type		Description
*	*	→	Si spécifié = objet est un nom d'objet (chaîne) Si omis = objet est un champ ou une variable
objet	Champ | Variable	→	Nom d'objet (si * spécifié) sinon Variable ou champ saisissable à sélectionner

Description

La commande ALLER A CHAMP permet de sélectionner l'objet saisissable objet (variable ou champ) en tant que zone active du formulaire. C'est l'équivalent d'un clic de l'utilisateur dans la zone ou de l'utilisation de la touche **Tabulation** pour sélectionner le champ ou la variable.

Si vous passez le paramètre optionnel *, vous indiquez que le second paramètre désigne le nom d'un objet (une chaîne). Si vous ne passez pas le paramètre *, vous indiquez que le paramètre désigne un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne de caractères mais la référence du champ ou de la variable (champs ou variables texte uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

Note : Cette commande fonctionne uniquement dans un contexte de saisie dans un formulaire. Elle ne fait rien lorsqu'elle est utilisée avec des zones de saisie situées dans un sous-formulaire en mode liste.

Exemples

(1) Voici les deux modes d'utilisation de la commande ALLER A CHAMP :

```
⇒ ALLER A CHAMP ([Personnel]Nom)  `Référence de champ
⇒ ALLER A CHAMP (*;"ZonePrénoms")  `Nom d'objet
```

(2) Reportez-vous à l'exemple de la commande REFUSER.

Référence

REFUSER.

REFUSER {(champ)}

Paramètre	Type		Description
champ	Champ	→	Champ dont la saisie doit être refusée

Description

REFUSER accepte deux syntaxes. Dans la première syntaxe, REFUSER n'a pas de paramètre. Dans ce cas, la commande rejette la totalité de la saisie et force l'utilisateur à rester dans le formulaire. La seconde syntaxe permet de ne refuser que champ et force l'utilisateur à rester dans le champ.

Note : Nous vous conseillons d'utiliser en priorité les outils intégrés de validation de saisie de 4D, avant de faire appel à cette commande.

La première syntaxe de REFUSER est utilisée pour empêcher l'utilisateur de valider un enregistrement incomplet. Vous pouvez parvenir au même résultat sans utiliser REFUSER : associez la touche **Entrée** à un bouton n'effectuant "Pas d'action" et utilisez les commandes VALIDER et NE PAS VALIDER pour valider ou annuler l'enregistrement, une fois que les champs ont été correctement remplis. Il est recommandé d'employer cette seconde technique plutôt que d'utiliser la première syntaxe de REFUSER.

En général, vous employez la première syntaxe de REFUSER pour empêcher l'utilisateur de valider un enregistrement incomplet ou comportant des valeurs incorrectes. Si l'utilisateur tente de valider l'enregistrement, l'exécution de REFUSER provoque l'annulation de cette commande et l'enregistrement reste affiché dans le formulaire. L'utilisateur doit alors recommencer la saisie jusqu'à ce que les valeurs soient considérées comme correctes ou annuler l'enregistrement.

Le meilleur emplacement pour la commande REFUSER, lorsque vous utilisez cette syntaxe, est la méthode objet d'un bouton de type Valider associé à la touche de validation. De cette manière, la validation n'est possible que lorsque l'enregistrement est accepté, et l'utilisateur ne peut pas "forcer" la validation en appuyant sur la touche **Entrée**.

La seconde syntaxe de REFUSER utilise le paramètre champ. Si elle est exécutée, le curseur reste dans la zone de saisie du champ. Cette syntaxe oblige l'utilisateur à saisir une valeur correcte. Cette instruction doit être appelée juste après la modification du champ. Vous pouvez tester la modification d'un champ à l'aide de la fonction Modifie.

Vous pouvez également placer la commande REFUSER dans la méthode objet de la zone de saisie. Lorsqu'elle est utilisée avec des champs de sous-formulaires, cette commande ne fait rien.

Vous devez placer cette syntaxe de REFUSER soit dans la méthode formulaire, soit dans une méthode objet du formulaire en train d'être modifié. Si vous utilisez REFUSER avec le formulaire "pleine page" d'un sous-formulaire, placez-la dans la méthode formulaire ou une méthode objet du formulaire "pleine page".

Vous pouvez utiliser la commande SELECTIONNER TEXTE pour sélectionner, à l'intérieur du champ, les valeurs qui ont été refusées.

Exemples

(1) L'exemple suivant illustre la première syntaxe de REFUSER, placée dans la méthode objet d'un bouton Valider. La touche **Entrée** a été définie comme équivalent clavier pour ce bouton. Cela signifie que même si l'utilisateur appuie sur cette touche pour valider l'enregistrement, la méthode objet du bouton sera exécutée. L'enregistrement est une transaction bancaire. Si la transaction est un chèque, un numéro de chèque doit être saisi. S'il n'y a pas de numéro, la validation est refusée :

```
   Au cas ou
         ` Si c'est un chèque sans numéro...
   : (([Opération]Trans = "Chèque") & ([Opération]Numéro = ""))
      ALERTE ("Veuillez saisir le numéro du chèque.") ` Alerter l'utilisateur
⇒     REFUSER ` Refuser la saisie
         ` Placer le curseur dans le champ "numéro de chèque"
      ALLER A CHAMP ([Opération]Numéro)
   Fin de cas
```

(2) L'exemple suivant est une partie de la méthode objet d'un champ [Employés]Salaire. La méthode objet teste si la valeur de ce champ est inférieure à 60 000 F et la refuse si c'est le cas. Vous pourriez effectuer le même contrôle en spécifiant une valeur minimum pour le champ, dans l'éditeur de formulaires du mode Structure :

```
   Si ([Employés]Salaire<60000)
      ALERTE ("Le salaire annuel doit être supérieur à 60 000 F.")
⇒     REFUSER ([Employés]Salaire)
   Fin de si
```

Référence

ALLER A CHAMP, NE PAS VALIDER, VALIDER.

23

Glisser-Déposer

Présentation du Glisser-Déposer

4e Dimension dispose de fonctions intégrées vous permettant de gérer le glisser-déposer ("drag and drop") parmi les objets de vos formulaires. Vous pouvez glisser-déposer un objet sur un autre objet situé dans la même fenêtre ou dans une autre fenêtre. Autrement dit, vous pouvez effectuer des glisser-déposer à l'intérieur d'un même process ou entre différents process.

4e Dimension ne comporte pas de fonctions intégrées permettant le glisser-déposer d'objets entre 4D et le bureau ou une autre application. Cependant, cette fonctionnalité est disponible via des plug-ins développés par les Partenaires 4D.

Note : Pour ne pas alourdir cette introduction, nous admettrons ici que le glisser-déposer permet de "transporter" des données d'un point à un autre. Nous verrons plus loin qu'un glisser-déposer peut aussi être la métaphore (c'est-à-dire la représentation au niveau de l'interface utilisateur) de toute opération, quelle qu'elle soit.

Propriétés d'objets "Glissable" et "Déposable"

Si vous souhaitez qu'un objet soit **glissable**, c'est-à-dire que vous puissiez le faire glisser et le déposer sur un autre objet, vous devez sélectionner la propriété "Glissable" pour cet objet dans la palette des propriétés d'objets. L'objet que vous faites glisser est appelé **objet source** de l'opération de glisser-déposer.

Si vous souhaitez qu'un objet soit **déposable**, c'est-à-dire que l'objet puisse être la destination d'une opération de glisser-déposer, vous devez sélectionner la propriété "Déposable" pour cet objet dans la palette des propriétés d'objets. L'objet qui reçoit les données est appelé **objet de destination** de l'opération de glisser-déposer.

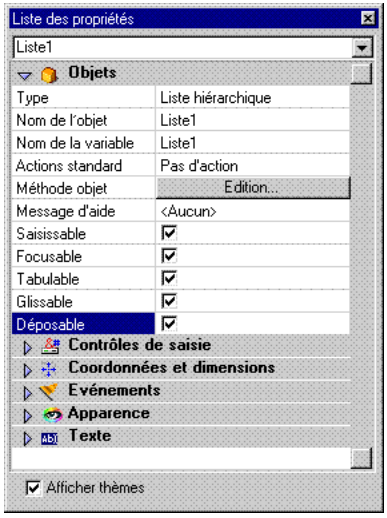
Par défaut, les objets nouvellement créés ne possèdent aucune de ces propriétés. Il est de votre ressort de les sélectionner explicitement.

Tous les objets situés dans un formulaire entrée ou dans une boîte de dialogue peuvent être définis comme glissables et déposables. Les éléments individuels d'un tableau (par exemple une zone de défilement) ou les éléments d'une liste hiérarchique peuvent être glissés et déposés. Inversement, vous pouvez faire glisser et déposer tout objet sur un élément individuel d'un tableau ou d'une liste hiérarchique. Il n'est toutefois pas possible de faire glisser et de déposer des objets depuis la zone de corps d'un formulaire sortie.

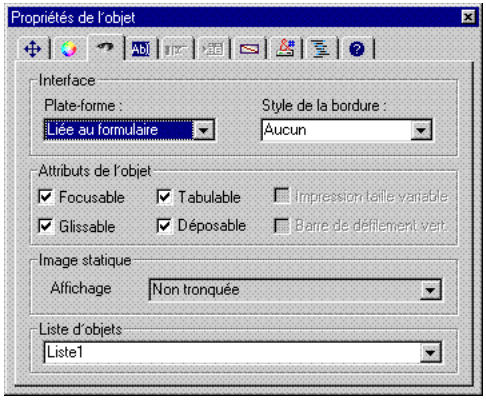
Afin de vous laisser "carte blanche" lors de la construction d'une interface utilisateur exploitant le glisser-déposer, 4D vous permet d'utiliser tout type d'objet actif (champ ou variable) en tant qu'objet source ou destination. Par exemple, si vous le souhaitez, vous pouvez glisser-déposer des boutons.

Notez qu'un objet "glissable" et "déposable" peut être déposé sur lui-même (à moins que vous n'interdisiez cette opération, reportez-vous aux paragraphes suivants).

Voici la Liste des propriétés, dans laquelle les propriétés "Glissable" et "Déposable" ont été définies pour l'objet sélectionné :

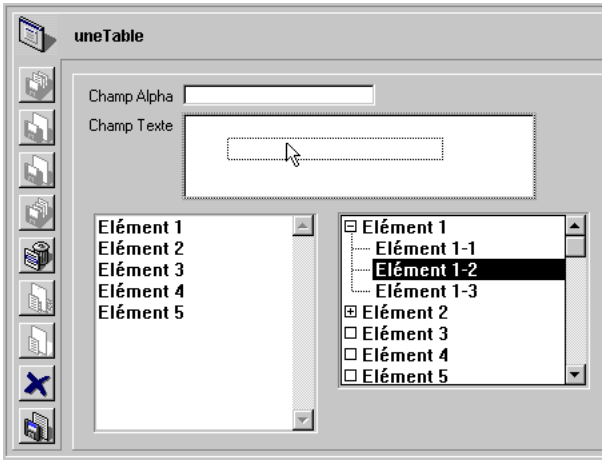


Ces propriétés peuvent également être définies dans la palette Propriétés de l'objet :



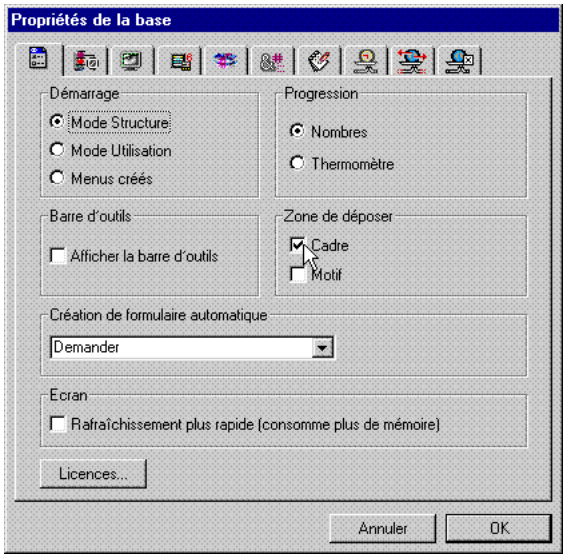
Gérer une interface utilisateur exploitant le glisser-déposer

4e Dimension gère l'aspect "interface utilisateur" du glisser-déposer. Si vous cliquez sur un objet glissable — et maintenez le bouton de la souris enfoncé — puis déplacez la souris, 4D représente à l'écran le déplacement de l'objet par un cadre pointillé suivant les mouvements de la souris. Dans l'écran ci-dessous, un élément de liste hiérarchique est sur le point d'être déposé dans un champ de type Texte :

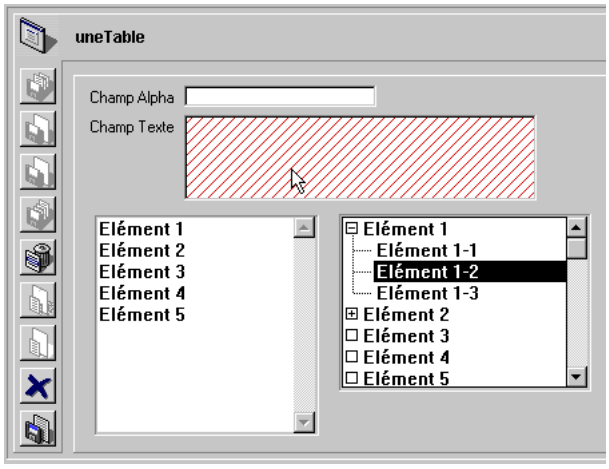


Notez le cadre grisé actif qui entoure alors le champ de texte. Cette activation indique l'objet de destination (ici le champ de texte). Si vous relâchez le bouton de la souris à cet instant, 4D comprend que vous voulez déposer l'objet que vous avez fait glisser sur l'objet activé.

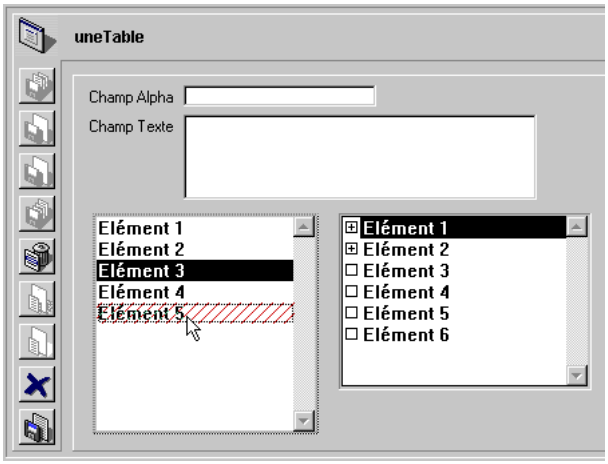
La représentation à l'écran de l'activation d'une zone de déposer peut être paramétrée dans la boîte de dialogue des Propriétés de la base :



Par défaut, le type d'activation est "Cadre". Cela correspond à l'encadrement de la zone tel que présenté dans l'écran précédent : un rectangle gris inversé apparaît autour de l'objet. Si vous utilisez des couleurs d'arrière-plan ou de contours d'objets s'accommodant mal de l'activation par encadrement, vous pouvez plutôt sélectionner le type d'activation "Motif". Dans ce cas, l'objet de destination est rempli par un motif de lignes diagonales, comme illustré ci-dessous :



Ici, un élément d'un tableau est déposé à l'intérieur du même tableau :



Vous pouvez, si vous le souhaitez, définir simultanément les deux types d'activation. Si vous n'en sélectionnez aucun, l'option par défaut sera "Cadre".

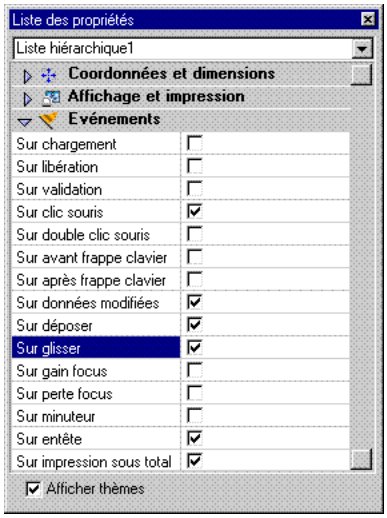
Note : L'activation de l'objet de destination “suit” chaque élément lorsque l'objet de destination est un tableau (zone de défilement) ou une liste hiérarchique.

Gérer le glisser-déposer par programmation

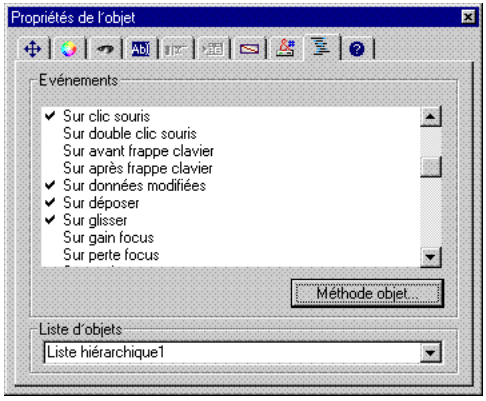
4e Dimension gère l'aspect "interface utilisateur" du glisser-déposer et il vous revient d'en traiter l'aspect "programmation". Pour cela, 4D vous fournit deux événements formulaires : Sur glisser et Sur déposer. **Ces deux événements sont envoyés à l'objet de destination**. Lors d'une opération de glisser-déposer, la méthode de l'objet source n'est sollicitée en aucun cas.

Pour que l'objet de destination puisse traiter ces deux événements, il doivent avoir été sélectionnés pour l'objet dans la Liste des propriétés ou dans la palette Propriétés de l'objet, comme illustré ci-desous :

• Liste des propriétés :



• Palette Propriétés de l'objet :



Sur glisser

L'événement Sur glisser est envoyé de manière répétée à l'objet de destination lorsque le pointeur de la souris est placé sur l'objet. Généralement, en réponse à cet événement, vous effectuez les actions suivantes :

• Vous appelez la commande PROPRIETES GLISSER DEPOSER qui vous renseigne sur l'objet source.
• En fonction de la nature et du type de l'objet de destination (celui duquel la méthode est en cours d'exécution) et de l'objet source, vous **acceptez** ou **refusez** le glisser.

Pour signaler que le glisser est accepté, la méthode de l'objet de destination doit retourner 0 (zéro), vous exécutez donc $0:=0. Pour signaler que le glisser est refusé, la méthode de l'objet de destination doit retourner -1 (moins un), vous exécutez donc $0:=-1. Pendant un événement Sur glisser, 4D traite la méthode de l'objet comme une fonction. Si aucun résultat n'est retourné, 4D considère que le glisser est accepté.

Si vous acceptez le glisser, l'objet de destination est activé. Si vous le refusez, l'objet de destination reste inactivé. Accepter le glisser ne signifie pas que les données glissées vont être insérées dans l'objet de destination. Cela signifie uniquement que l'objet de destination, si le bouton de la souris était relâché à cet instant, accepterait les données.

Si vous ne traitez pas l'événement Sur glisser pour un objet dont la propriété "Déposable" a été sélectionnée, l'objet sera activé pour tous les glisser, quelle que soit la nature et le type des données glissées.

La traitement de l'événement Sur glisser vous permet de contrôler la première phase d'une opération de glisser-déposer : non seulement vous pouvez tester si le type des données glissées est compatible avec l'objet de destination — et donc accepter ou refuser le glisser — mais également, vous en informez l'utilisateur, car 4D active ou non l'objet de destination en fonction de votre décision.

Le code traitant un événement Sur glisser doit être court et s'exécuter rapidement car cet événement est envoyé de manière répétée à l'objet de destination courant, en fonction des mouvements de la souris.

ATTENTION : Si le glisser-déposer est un **glisser-déposer interprocess**, ce qui signifie que l'objet source est situé dans un process (fenêtre) différent de celui de l'objet de destination, la méthode de l'objet de destination lors de l'événement Sur glisser est exécutée **dans le contexte du process source** (le process de l'objet source) et non dans le process de l'objet de destination. C'est le seul cas où un tel type d'exécution a lieu (les avantages que procure ce fonctionnement sont décrits à la fin de cette section).

Sur déposer

L'événement Sur déposer est envoyé (une seule fois) à l'objet de destination lorsque le bouton de la souris est relâché alors que le pointeur se trouvait au-dessus de l'objet. Cet événement est la seconde phase d'un glisser-déposer, dans laquelle vous effectuez les véritables opérations répondant à l'action de l'utilisateur.

Cet événement n'est pas envoyé à l'objet si le glisser n'a pas été accepté dans le ou les événement(s) Sur glisser. Si vous traitez l'événement Sur glisser pour un objet et refusez le glisser, l'événement Sur déposer ne se déclenche pas. Ainsi, si pendant l'événement Sur glisser vous avez testé la compatibilité entre le type des données de l'objet source et de destination et accepté le glisser, vous n'avez pas besoin de tester de nouveau les données dans l'événement Sur déposer : vous savez déjà qu'elles sont compatibles.

L'aspect le plus intéressant de l'implémentation du glisser-déposer dans 4D est que le programme vous permet de faire ce que vous voulez. Par exemple :

• Si un élément de liste hiérarchique est déposé sur un champ de type Texte, vous pouvez décider d'insérer le texte de l'élément de liste au début, au milieu ou à la fin du champ de texte.
• Votre formulaire contient un bouton image à deux états représentant une corbeille vide et une corbeille pleine. Le glisser-déposer d'un objet sur ce bouton pourrait provoquer, du point de vue de l'interface utilisateur : “supprimer l'objet qui a été glissé-déposé dans la corbeille”. Ici, le glisser-déposer ne transporte pas des données d'un point à un autre, mais plutôt il déclenche une action.
• Glisser un élément de tableau depuis une palette flottante vers un objet dans un formulaire pourrait signifier “afficher dans cette fenêtre l'enregistrement du client dont le nom a été glissé-déposé depuis la fenêtre flottante listant les noms de tous les clients stockés dans la base”.
• Etc.

La gestion du glisser-déposer de 4D est une boîte à outils vous permettant d'implémenter toutes les métaphores d'interface utilisateur auxquelles vous pouvez penser.

Les commandes du thème Glisser-Déposer

La commande PROPRIETES GLISSER DEPOSER retourne :
• un pointeur vers l'objet glissé (champ ou variable),
• le numéro de l'élément de tableau ou de liste hiérarchique le cas échéant,
• le numéro du process source.

La commande Position deposer retourne le numéro ou la position de l'élément cible si l'objet de destination est un tableau (c'est-à-dire une zone de défilement) ou une liste hiérarchique.

Les commandes telles que RESOUDRE POINTEUR et Type sont utiles pour tester la nature et le type de l'objet source.

Lorsque l'opération de glisser-déposer est destinée à copier les données glissées :
• Si le glisser-déposer est effectué à l'intérieur du même process, utilisez ces commandes pour effectuer les actions correspondantes (c'est-à-dire simplement assigner l'objet source à l'objet de destination).
• Si le glisser-déposer est interprocess, soyez vigilant lorsque vous accédez aux données glissées : vous devez récupérer l'instance des données située dans le process source. Si les données glissées proviennent d'une variable, utilisez la commande LIRE VARIABLE PROCESS pour obtenir la valeur correcte. Si les données glissées proviennent d'un champ, il est probable que l'enregistrement courant de la table n'est pas le même d'un process à l'autre, vous devez donc accéder au bon enregistrement.

Dans ce dernier cas, plusieurs solutions sont envisageables :
• Puisque l'événement Sur glisser de la méthode de l'objet de destination est exécuté dans le contexte du process source, vous pouvez copier, au moment de son exécution, les données du champ ou le numéro de l'enregistrement dans une variable interprocess, que vous réutiliserez lors de l'événement Sur déposer.
• Pendant l'événement Sur déposer, vous pouvez établir une communication interprocess avec le process source dans le but de récupérer les données requises.

Lorsque le glisser-déposer n'est pas destiné à déplacer des données mais plutôt à créer des métaphores d'interface pour des opérations particulières, vous pouvez virtuellement réaliser tout ce que vous voulez.

Référence

Evenement formulaire, LIRE VARIABLE PROCESS, Liste existante, Position deposer, PROPRIETES GLISSER DEPOSER, RESOUDRE POINTEUR, Type.

Position deposer

Position deposer → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Numéro (tableau) ou position (liste) de l'élément de destination ou -1 si le déposer a lieu après le dernier élément de tableau ou de liste

Description

Position deposer retourne le numéro de l'élément de tableau ou la position de l'élément de liste hiérarchique sur lequel un objet a été (glissé et) déposé.

Généralement, vous utiliserez Position deposer pendant le traitement d'un événement glisser-déposer qui s'est produit dans un tableau ou une liste hiérarchique.

Si l'objet de destination est un tableau, la fonction retourne un numéro d'élément. Si l'objet de destination est une liste hiérarchique, la fonction retourne une position d'élément. Dans les deux cas, la fonction retourne -1 si l'objet source a été déposé après le dernier élément du tableau ou de la liste.

Si vous appelez Position deposer pendant le traitement d'un événement qui n'est pas de type glisser-déposer dans un tableau ou dans une liste, la fonction retourne également -1.

Rappel : Pour qu'un objet de formulaire accepte des données déposées, la propriété **Déposable** doit lui avoir été assignée. De plus, sa méthode objet doit être appelée par l'événement Sur glisser et/ou Sur déposer si vous voulez pouvoir gérer ce type d'événement.

Exemples

Reportez-vous à l'exemple de la commande PROPRIETES GLISSER DEPOSER.

Référence

Présentation du Glisser-Déposer, PROPRIETES GLISSER DEPOSER.

PROPRIETES GLISSER DEPOSER (srcObjet; srcElément; srcProcess)

Paramètre	Type		Description
srcObjet	Pointeur	←	Pointeur vers l'objet source du glisser-déposer
srcElément	Numérique	←	Numéro de l'élément de tableau glissé ou Elément de la liste hiérarchique glissé ou -1 si l'objet glissé n'est pas un élément de tableau ni de liste
srcProcess	Numérique	←	Numéro du process source

Description

La commande PROPRIETES GLISSER DEPOSER vous permet de récupérer des informations sur l'objet source lorsque l'événement Sur glisser ou Sur déposer est déclenché pour un objet.

Généralement, la commande PROPRIETES GLISSER DEPOSER se place dans la méthode objet (ou une des sous-méthodes qu'elle appelle) de l'objet pour lequel l'événement Sur glisser ou Sur déposer se produit.

Rappel : Des données peuvent être déposées sur un objet de formulaire si la propriété **Déposable** lui a été assignée. De plus, la méthode qui lui est associée doit être appelée par l'événement Sur déposer et/ou Sur glisser si vous voulez traiter ce type d'événement.

Après l'appel de cette commande :

• Le paramètre srcObjet est un pointeur vers l'objet source, c'est-à-dire l'objet qui a été glissé et déposé. Notez que cet objet peut être ou non identique à l'objet de destination, autrement dit l'objet pour lequel l'événement Sur déposer ou Sur glisser a été déclenché. Le glisser-déposer de valeurs entre des objets de même type est utile pour les tableaux et les listes hiérarchiques : cela vous fournit un moyen simple de permettre à l'utilisateur de trier manuellement un tableau ou une énumération.

• Si les données glissées-déposées sont un élément de tableau (l'objet source étant un tableau), le paramètre srcElément est égal au numéro de cet élément. Si les données glissées-déposées sont un élément de liste hiérarchique, le paramètre srcElément retourne la position de cet élément. Sinon, si l'objet source n'est ni un tableau ni une liste hiérarchique, srcElément est égal à -1.

• Des opérations de glisser-déposer peuvent être effectuées entre différents process. Le paramètre srcProcess est égal au numéro du process auquel appartient l'objet source. Il est important de tester la valeur de ce paramètre. En effet, vous pouvez traiter un glisser-déposer “process” en copiant simplement les données source dans l'objet de destination. En revanche, lorsque vous traitez un glisser-déposer “interprocess”, vous devez utiliser la commande LIRE VARIABLE PROCESS pour récupérer les données source à partir de l'instance de l'objet du process source. Si l'objet source est un champ, vous devez récupérer sa valeur dans le process source via les outils de communication interprocess ou traiter ce cas particulier pendant que vous répondez à l'événement Sur glisser (reportez-vous aux paragraphes ci-dessous). Notez cependant que généralement, le glisser-déposer dans une interface utilisateur s'effectue à partir de variables (tableaux et liste hiérarchiques) vers des zones de saisie de données (champs ou variables).

Si vous appelez PROPRIETES GLISSER DEPOSER alors qu'aucun événement glisser-déposer ne s'est produit, srcObjet retourne un pointeur NIL, srcElément retourne -1 et srcProcess retourne 0.

Astuce : 4e Dimension gère pour vous l'aspect graphique du glisser-déposer. Mais c'est à vous de traiter l'événement de manière appropriée. Dans les exemples ci-dessous, le traitement consiste à copier les données qui ont été glissées. Mais vous pouvez également implémenter des interfaces plus sophistiquées dans lesquelles, par exemple, le glisser-déposer d'un élément de tableau depuis une palette flottante provoque le remplissage de la fenêtre de destination (la fenêtre dans laquelle se trouve l'objet de destination) avec des données structurées (comme plusieurs champs provenant d'un enregistrement désigné par l'élément de tableau source).

Vous pouvez appeler la commande PROPRIETES GLISSER DEPOSER lors de l'événement formulaire Sur glisser afin de décider si l'objet de destination doit ou non accepter l'opération, en fonction du type et/ou de la nature de l'objet source (ou pour toute autre raison). Si vous acceptez le glisser-déposer, la méthode de l'objet doit retourner $0:=0. Si vous n'acceptez pas l'opération, la méthode de l'objet doit retourner $0:=-1. L'acceptation ou le refus d'un glisser-déposer est visible à l'écran : l'objet sera ou ne sera pas sélectionnable (par exemple encadré) en tant que destination du glisser-déposer.

Astuce : Pendant l'événement Sur glisser, la méthode de l'objet de destination est exceptionnellement exécutée dans le contexte du process de l'objet source. Si le glisser-déposer est interprocess et si l'objet source est un champ, vous pouvez profiter de l'occasion pour copier les données source dans une variable interprocess. Ainsi, vous n'aurez pas besoin par la suite, pendant l'événement Sur déposer, d'ouvrir une communication interprocess avec le process source pour récupérer la valeur du champ glissé. Si l'objet source du glisser-déposer interprocess est une variable, vous pouvez utiliser la commande LIRE VARIABLE PROCESS pendant l'événement Sur déposer.

Exemples

(1) Vous disposez, dans plusieurs formulaires de votre base, de zones de défilement. Vous voulez que l'utilisateur puisse réordonner manuellement les éléments des zones par simple glisser-déposer à l'intérieur de chaque zone. Plutôt que d'écrire du code spécifique pour chaque cas, vous voulez utiliser une méthode projet générique qui traite toutes les zones de défilement. Pour cela, vous pouvez écrire :

```
` Méthode projet de traitement de glisser-déposer interne dans un tableau
` Traitement de glisser-déposer interne dans un tableau ( Pointeur ) -> Booléen
` Traitement de glisser-déposer interne dans un tableau ( -> Tableau )
                                        -> Est un glisser-déposer interne dans un tableau

   Au cas ou
      : (Evenement formulaire=Sur glisser)
⇒        PROPRIETES GLISSER DEPOSER ($vpSrcObj;$vlSrcElem;$vlPID)
         Si ($vpSrcObj=$1)
               `Accepter le glisser-déposer s'il est interne au tableau
            $0:=0
         Sinon
            $0:=-1
         Fin de si
      : (Evenement formulaire=Sur déposer)
            ` Récupérer les informations sur l'objet source du glisser-déposer
⇒        PROPRIETES GLISSER DEPOSER ($vpSrcObj;$vlSrcElem;$vlPID)
            ` Récupérer le numéro de l'élément de destination
         $vlDstElem := Position deposer
            ` Si l'élément n'a pas été glissé-déposé sur lui-même
         Si ($vlDstElem # $vlSrcElem)
               ` Stocker l'élément glissé dans l'élément 0 du tableau
            $1->{0}:=$1->{$vlSrcElem}
               ` Effacer l'élément glissé
            SUPPRIMER LIGNES ($1->;$vlSrcElem)
               ` Si l'élément de destination est au-delà de l'élément glissé
            Si ($vlDstElem>$vlSrcElem)
                  ` Décrémenter le numéro de l'élément de destination
               $vlDstElem:=$vlDstElem-1
            Fin de si
                  ` Si le glisser-déposer s'est produit au-delà du dernier élément
            Si ($vlDstElem=-1)
                  ` Définir le numéro de l'élément de destination comme un nouvel
                  ` élément ajouté à la fin du tableau
               $vlDstElem:=Taille tableau ($1->)+1
            Fin de si
```

```
            ` Insérer ce nouvel élément
         INSERER LIGNES ($1->;$vlDstElem)
            ` Fixer sa valeur, préalablement stockée dans l'élément zéro du tableau
         $1->{$vlDstElem}:=$1->{0}
            ` L'élément devient le nouvel élément sélectionné du tableau
         $1->:=$vlDstElem
      Fin de si
   Fin de cas
```

Une fois que vous avez écrit cette méthode projet, vous pouvez l'utiliser ainsi :

```
   ` Méthode objet Zone de défilement unTableau

   Au cas ou
         `...
      : (Evenement formulaire=Sur glisser )
         $0:=Traitement de glisser-déposer interne dans un tableau (Self)
      : (Evenement formulaire=Sur déposer )
         Traitement de glisser-déposer interne dans un tableau (Self))
         `...
   Fin de cas
```

(2) Vous disposez, dans plusieurs formulaires de votre base, de zones de texte saisissables. Vous voulez que l'utilisateur puisse y saisir des données par glisser-déposer à partir de sources multiples. Plutôt que d'écrire du code spécifique pour chaque cas, vous voulez utiliser une méthode projet générique qui traite toutes les zones de texte. Pour cela, écrivez la méthode suivante :

```
   ` Méthode projet Traitement du déposer dans variable Texte
   ` Traitement du déposer dans variable Texte ( Pointeur )
   ` Traitement du déposer dans variable Texte ( -> variable texte ou chaîne )

   Au cas ou
      ` Utilisation de cet événement pour accepter ou refuser le glisser-déposer
      : (Evenement formulaire=Sur glisser)
            ` Initialiser $0 pour le refus
         $0:=-1
            ` Récupérer les informations sur l'objet source du glisser-déposer
⇒        PROPRIETES GLISSER DEPOSER($vpSrcObj;$vlSrcElem;$vlPID)
            ` Dans cet exemple, nous refusons le glisser-déposer d'un objet sur lui-même
         Si ($vpSrcObj#$1)
               ` Récupérer le type des données glissées
            $vlSrcType:=Type($vpSrcObj->)
```

```
      Au cas ou
         : ($vlSrcType=Est un champ alpha)
               ` OK pour les champs alphanumériques
            $0:=0
               ` Copie immédiate de la valeur dans une variable interprocess
            <>vtDonnéesGlissées:=$vpSrcObj->
         : ($vlSrcType=Est un texte)
               ` OK pour les champs ou variables texte
            $0:=0
            RESOUDRE POINTEUR($vpSrcObj;$vsVarName;$vlTableNum;
                                                              $vlFieldNum)
               ` Si c'est un champ
            Si (($vlTableNum>0) & ($vlFieldNum>0))
               ` Copie immédiate de la valeur dans une variable interprocess
               <>vtDonnéesGlissées:=$vpSrcObj->
            Fin de si
         : ($vlSrcType=Est une variable chaîne)
               ` OK pour les variables chaîne
            $0:=0
         : (($vlSrcType=Est un tableau chaîne) | ($vlSrcType=Est un tableau texte))
               ` OK pour les tableaux chaîne et texte
            $0:=0
         : (($vlSrcType=Est un entier long) | ($vlSrcType=Est un numérique))
            Si (Liste existante($vpSrcObj->))
               ` OK pour les liste hiérarchiques
               $0:=0
            Fin de si
      Fin de cas
   Fin de si

      ` Utilisation de cet événement pour effectuer l'action de glisser-déposer
: (Evenement formulaire=Sur déposer)
   $vtDonnéesGlissées:=""
      ` Récupérer les informations sur l'objet source du glisser-déposer
⇒  PROPRIETES GLISSER DEPOSER($vpSrcObj;$vlSrcElem;$vlPID)
   RESOUDRE POINTEUR($vpSrcObj;$vsVarName;$vlTableNum;$vlFieldNum)
      `Si c'est un champ
   Si (($vlTableNum>0) & ($vlFieldNum>0)
         ` Récupérons la variable interprocess créée lors du Sur glisser
      $vtDonnéesGlissées:=<>vtDonnéesGlissées
   Sinon
         ` Récupérer le type des données glissées
      $vlSrcType:=Type($vpSrcObj->)
```

```
   Au cas ou
         ` Si c'est un tableau
      : (( $vlSrcType=Tableau chaîne) | ($vlSrcType=Tableau texte))
         Si ($vlPID#Numero du process courant)
               ` Lire l'élément depuis l'instance de la variable dans le process
               ` source
            LIRE VARIABLE PROCESS ($vlPID;$vpSrcObj->{$vlSrcElem};
                                                            $vtDraggedData)
         Sinon
               ` Glisser-déposer depuis le même process, copions juste la valeur

            $vtDraggedData:=$vpSrcObj->{$vlSrcElem}
         Fin de si
      : (($vlSrcType=Est un numérique ) | ($vlSrcType=Est un entier long )
            ` Si c'est une liste hiérarchique
         Si (Liste existante($vpSrcObj->))
                  ` Est-ce une liste d'un autre process ?
            Si ($vlPID#Numero du process courant)
                  `Récupérer la référence de la liste dans l'autre process
               LIRE VARIABLE PROCESS ($vlPID;$vpSrcObj->;$vList)
            Sinon
               $vList:=$vpSrcObj->
            Fin de si
               `Récupérer le texte de l'élément dont on a obtenu la position
            INFORMATION ELEMENT($vlList;$vlSrcElem;$vlItemRef;
                                                            $vsItemText)
            $vtDraggedData:=$vsItemText
         Fin de si
   Sinon
         ` C'est une variable chaîne ou texte
         Si ($vlPID#Numero du process courant)
            LIRE VARIABLE PROCESS ($vlPID;$vpSrcObj->;$vtDraggedData)
         Sinon
         $vtDraggedData:=$vpSrcObj->
         Fin de si
   Fin de cas
Fin de si
   ` S'il y a effectivement quelque chose à déposer (l'objet source peut être vide)
Si ($vtDraggedData # "")
   ` Vérifions que la longueur de la variable texte ne dépasse pas
   ` 32 000 caractères
   Si ((Longueur($1->)+Longueur($vtDraggedData))<=32000)
      $1->:=$1->+$vtDraggedData
```

```
            Sinon
               BEEP
               ALERTE("Le glisser-déposer ne peut être effectué car il y aurait
                                                                    trop de texte.")
            Fin de si
         Fin de si
      Fin de cas
```

Une fois que vous avez écrit cette méthode projet, vous pouvez l'utiliser ainsi :

```
` Méthode objet du champ de texte [uneTable]unTexte

Au cas ou
   ` ...
: (Evenement formulaire=Sur glisser )
   $0:=Traitement du déposer dans variable texte  (Self)

: (Evenement formulaire=Sur déposer )
   Traitement du déposer dans variable texte  (Self)
   ` ...

Fin de cas
```

Référence

Evenement formulaire, LIRE VARIABLE PROCESS, Liste existante, Position deposer, Présentation du Glisser-Déposer, RESOUDRE POINTEUR.

24

Graphes

version 6.0 (Modifiée)

Notes version 6 : A partir de la version 6 de 4e Dimension, les graphes sont réalisés à l'aide du plug-in 4D Chart, intégré à 4e Dimension. Les commandes du thème **Graphe** de la version précédente de 4D sont désormais redirigées de manière transparente vers 4D Chart.
• Si vous souhaitez personnaliser une Zone de graphe placée dans un formulaire avec les commandes de 4D Chart, utilisez le paramètre graphZone décrit dans cette commande comme référence de zone externe pour les commandes de 4D Chart. Pour plus d'informations sur les commandes de 4D Chart, reportez-vous au manuel *Langage* de 4D Chart.
• La commande GRAPHE fonctionne avec une Zone de graphe créée dans un formulaire 4D. Elle doit impérativement être placée dans la méthode formulaire ou dans une méthode objet appartenant au formulaire, ou encore dans une méthode projet appelée par l'une des deux précédentes.

GRAPHE (graphZone; graphNum; xCatégories; zValeurs{; zValeurs2; ...; zValeursN})

Paramètre	Type		Description
graphZone	Variable	→	Zone de graphe dans le formulaire
graphNum	Numérique	→	Numéro de type de graphe
xCatégories	Tableau | Sous-champ	→	Catégories sur l'axe des x
zValeurs	Tableau | Sous-champ	→	Valeurs à représenter graphiquement (jusqu'à 8 valeurs)

Description

La commande GRAPHE crée un graphe dans une zone de graphe placée dans un formulaire. Les valeurs à représenter peuvent provenir soit de tableaux soit de sous-champs.

Le paramètre graphZone est le nom de la zone de graphe qui affiche le graphe. Les zones de graphe sont créées en mode Structure, dans l'éditeur de formulaires. Le nom du graphe est celui de la variable de type Zone de graphe. Pour plus d'informations sur la création de zones de graphe dans les formulaires, reportez-vous au manuel *Mode Structure* de 4e Dimension.

Le paramètre graphNum définit le type de graphe à utiliser. Vous devez passer un nombre entre 1 et 8. Les différents types de graphes disponibles sont listés dans l'exemple présenté plus bas. Une fois le graphe créé, vous pouvez modifier son type en modifiant la valeur de graphNum et en exécutant de nouveau la commande GRAPHE.

Le paramètre xCatégories définit les valeurs qui seront utilisées pour l'axe des X. Vous pouvez passer un champ de type Alpha, Heure, Date, ou un type numérique. Il doit y avoir le même nombre de sous-enregistrements ou d'éléments de tableau dans xCatégories qu'il y en a dans chaque zValeurs.

Le paramètre zValeurs définit les valeurs à représenter graphiquement. Elles doivent être de type numérique. Vous pouvez passer jusqu'à huit ensembles de données. Les graphes en secteurs ne représentent que le premier zValeurs.

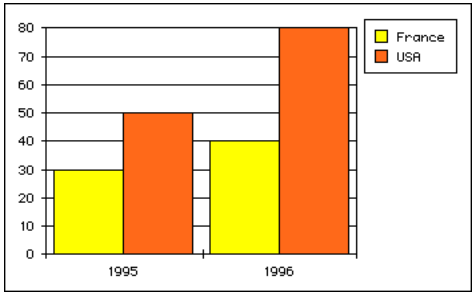
Exemples

(1) L'exemple suivant illustre l'utilisation des tableaux pour générer un graphe. Ce code doit être placé dans la méthode formulaire (ou une méthode objet) du formulaire contenant la zone de graphe. A noter que, dans notre exemple, les données représentées sont constantes, ce qui n'est généralement pas le cas :

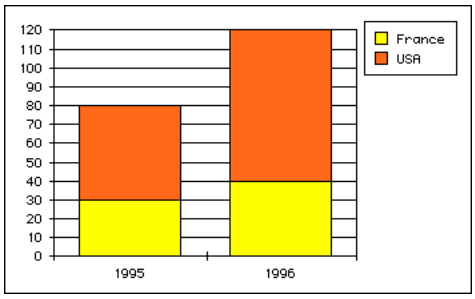
```
   TABLEAU ALPHA (4; X; 2)  ` Création d'un tableau pour l'axe des X
   X{1} := "1995"  ` libellé X #1
   X{2} := "1996"  ` Libellé X #2
   TABLEAU REEL (A; 2)  ` Création d'un tableau pour l'axe des Z
   A{1} := 30  ` Insertion des données
   A{2} := 40
   TABLEAU REEL (B; 2) ` Création d'un tableau pour l'axe des Z
   B{1} := 50  ` Insertion des données
   B{2} := 80
⇒  GRAPHE (vGraph; vType; X; A; B) ` Dessiner le graphe
      ` Définition des légendes du graphe
   PARAMETRES DU GRAPHE (vGraph; 0; 0; 0; 0; Faux; Faux; Vrai; "France"; "USA")
```

Les images suivantes représentent les graphes résultants :

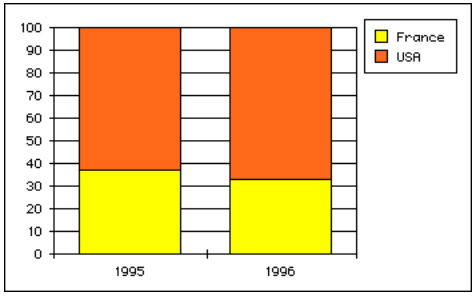
• Lorsque vType est égal à 1, vous obtenez un graphe en **Colonnes** :



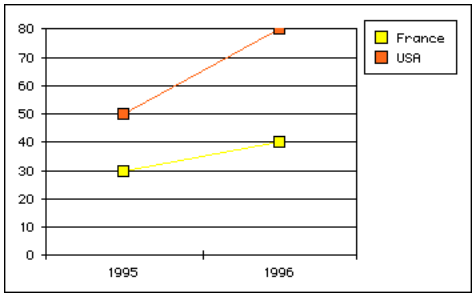
• Lorsque vType est égal à 2, vous obtenez un graphe en **Colonnes proportionnelles** :



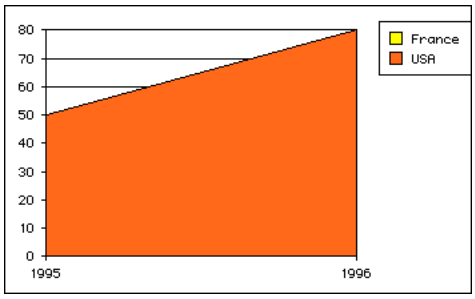
• Lorsque vType est égal à 3, vous obtenez un graphe en **Colonnes empilées** :



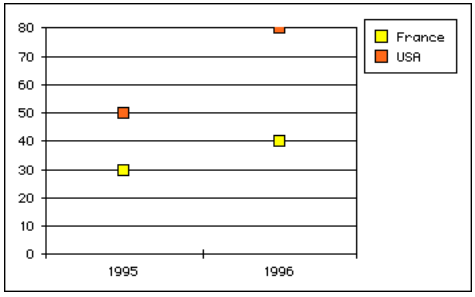
• Lorsque vType est égal à 4, vous obtenez un graphe en **Lignes** :



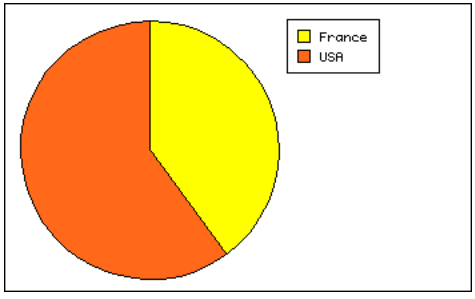
• Lorsque vType est égal à 5, vous obtenez un graphe en **Aires** :



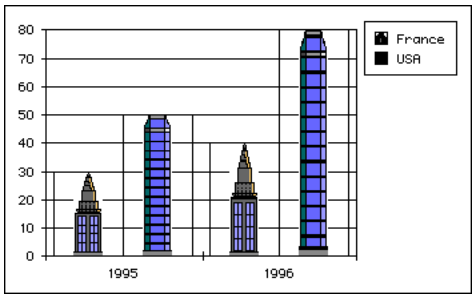
• Lorsque vType est égal à 6, vous obtenez un graphe en **Points** :



• Lorsque vType est égal à 7, vous obtenez un graphe en **Secteurs** :



• Lorsque vType est égal à 8, vous obtenez un graphe en **Images** :



(2) L'exemple suivant effectue la représentation graphique des ventes des commerciaux. La sous-table comporte trois champs : Nom, TotalAnnéePrécédente, et TotalCetteAnnée. Le graphe affichera les ventes de chaque commercial pour les deux dernières années :

⇒ **GRAPHE** (GrapheVentes; 1; Commerciaux'Nom; Commerciaux'TotalAnnéePrécédente; Commerciaux'TotalCetteAnnée)

Référence

GRAPHE SUR SELECTION, PARAMETRES DU GRAPHE.

PARAMETRES DU GRAPHE

Graphes

version 6.0 (Modifiée)

Notes version 6 : A partir de la version 6 de 4e Dimension, les graphes sont réalisés à l'aide du plug-in 4D Chart, intégré à 4e Dimension. Les commandes du thème **Graphe** de la version précédente de 4D sont désormais redirigées de manière transparente vers 4D Chart.

• Si vous souhaitez personnaliser une Zone de graphe placée dans un formulaire avec les commandes de 4D Chart, utilisez le paramètre grapheZone décrit dans cette commande comme référence de zone externe pour les commandes de 4D Chart. Pour plus d'informations sur les commandes de 4D Chart, reportez-vous au manuel *Langage* de 4D Chart.

• La commande PARAMETRES DU GRAPHE fonctionne avec une Zone de graphe créée dans un formulaire 4D. Elle doit impérativement être placée dans la méthode formulaire ou dans une méthode objet appartenant au formulaire, ou encore dans une méthode projet appelée par l'une des deux précédentes.

PARAMETRES DU GRAPHE (grapheZone; xmin; xmax; ymin; ymax; xprop; grilleX; grilleY; titre{; titre2; ...; titreN})

Paramètre	Type		Description
grapheZone	Variable	→	Nom de la zone de graphe
xmin	Numérique, Date ou Heure	→	Valeur minimale de l'échelle des X pour graphe proportionnel (lignes ou points)
xmax	Numérique, Date ou Heure	→	Valeur maximale de l'échelle des X pour graphe proportionnel (lignes ou points)
ymin	Numérique	→	Valeur minimale de l'échelle des Y
ymax	Numérique	→	Valeur maximale de l'échelle des Y
xprop	Booléen	→	VRAI pour l'échelle des X proportionnelle ; FAUX pour l'échelle des X normale (lignes ou points)
grilleX	Booléen	→	VRAI pour la grille sur l'axe des X ; FAUX pour pas de grille sur l'axe des X (seulement si xprop est VRAI)
grilleY	Booléen	→	VRAI pour la grille sur l'axe des Y; FAUX pour pas de grille sur l'axe des Y
titre	Alpha	→	Titre(s) pour les titre(s) des série(s)

Description

La commande PARAMETRES DU GRAPHE permet de paramétrer les échelles et les grilles d'un graphe placé dans un formulaire. Le graphe doit déjà être affiché à l'aide de la commande GRAPHE. PARAMETRES DU GRAPHE ne fait rien s'il s'agit d'un graphe de type secteurs.

Les paramètres xmin, xmax, ymin et ymax fixent les valeurs minimales et maximales pour les axes des X ou Y. Si la valeur des deux paramètres correspondants au même axe est nulle (0, ?00:00:00? ou !00/00/00! selon le type de données), les valeurs de graphe par défaut seront utilisées.
Le paramètre xprop fixe l'axe des X comme proportionnel (sont concernés par cette option les graphes de type 4 et 6). Lorsque ce paramètre est Vrai, chaque point sera mis sur l'axe des X par rapport aux valeurs des points si elles sont de type numérique, heure ou date.

Les paramètres grilleX et grilleY montrent ou cachent les grilles. Une grille pour l'axe des X sera affichée s'il s'agit d'un graphe en points ou en lignes proportionnel.

Le(s) paramètre(s) titre spécifient les titres des légendes.

Exemple

Reportez-vous à l'exemple de la commande GRAPHE.

Référence

GRAPHE, GRAPHE SUR SELECTION.

GRAPHE SUR SELECTION

Graphes

version 6.0 (Modifiée)

Note version 6 : A partir de la version 6 de 4e Dimension, les graphes sont réalisés à l'aide du plug-in 4D Chart, intégré à 4e Dimension. Les commandes du thème **Graphe** de la version précédente de 4D sont désormais redirigées de manière transparente vers 4D Chart. Pour plus d'informations sur les commandes de 4D Chart, reportez-vous au manuel *Langage* de 4D Chart.

GRAPHE SUR SELECTION{(table)}

ou :

GRAPHE SUR SELECTION ({table; }numGraphe; axeX; axeZ{; axeZ2; ...; axeZN})

Paramètre	Type		Description
table	Table	→	Table de la sélection à représenter graphiquement ou Table par défaut si ce paramètre est omis
numGraphe	Numérique	→	Numéro de type de graphe
axeX	Champ	→	Valeurs sur l'axe des X
axeZ	Champ	→	Champ(s) à représenter (jusqu'à 8 champs)

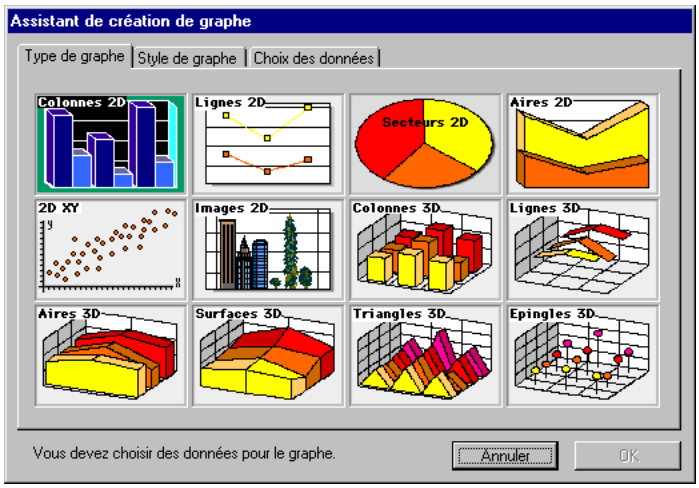
Description

GRAPHE SUR SELECTION accepte deux syntaxes. La première syntaxe fait apparaître l'Assistant de création de graphe, permettant à l'utilisateur de sélectionner les champs à représenter graphiquement. Dans la seconde syntaxe, vous désignez les champs à représenter et l'Assistant de création de graphe ne s'affiche pas.

GRAPHE SUR SELECTION ne traite que les valeurs des champs d'une table. Seules les valeurs de la sélection courante du process courant sont représentées.

La première syntaxe est équivalente à la sélection de la commande **Graphes...** dans le menu **Etats**, en mode Utilisation.

L'écran suivant représente la première page de l'assistant de création de graphe, permettant à l'utilisateur de choisir un type de graphe.



Pour plus d'informations sur l'emploi de cet assistant, reportez-vous au manuel *Mode Utilisation*.

La seconde syntaxe de la commande représente graphiquement les champs spécifiés de table.

Le paramètre numGraphe définit le type de graphe à utiliser. Vous devez passer un nombre entre 1 et 8. Les différents types de graphes disponibles sont représentés dans l'exemple de la commande GRAPHE.

Le paramètre axeX définit les valeurs qui seront utilisées pour l'axe des X. Vous pouvez passer un champ de type Alpha, Entier, Entier long, Numérique ou Date.

Le paramètre axeZ définit les valeurs à représenter graphiquement. Vous pouvez passer un champ de type Entier, Entier long ou Numérique. Vous pouvez passer jusqu'à huit champs dans axeZ, séparés par deux points.

Quelle que soit la syntaxe employée, GRAPHE SUR SELECTION crée une fenêtre 4D Chart dans laquelle vous travaillez avec les graphes que vous créez.

Note : Vous pouvez également créer des graphes avec 4D dans l'Editeur d'états semi-automatiques, en demandant une impression vers un graphe. Pour plus d'informations sur ce point, reportez-vous au manuel *Mode Utilisation* de 4e Dimension.

Exemples

(1) L'exemple suivant illustre l'utilisation de la première syntaxe de GRAPHE SUR SELECTION. La fenêtre de l'Assistant de création de graphe s'affiche, permettant à l'utilisateur de sélectionner un type de graphe. La première méthode permet à l'utilisateur d'effectuer une recherche dans la table [Personnes], puis d'effectuer un tri ; ensuite, l'Assistant de création de graphes est affiché :

```
CHERCHER ([Personnes])
Si(OK=1)
   TRIER ([Personnes])
   Si(OK=1)
⇒     GRAPHE SUR SELECTION([Personnes])
   Fin de si
Fin de si
```

(2) L'exemple suivant illustre l'utilisation de la seconde syntaxe de GRAPHE SUR SELECTION. Une recherche puis un tri sont d'abord effectués sur les enregistrements de la table [Personnes]. Les salaires des personnes sont alors représentés graphiquement :

```
CHERCHER ([Personnes]; [Personnes]Titre = "Directeur")
TRIER ([Personnes]; [Personnes]Salaire; >)
⇒ GRAPHE SUR SELECTION([Personnes]; 1; [Personnes]Noms; [Personnes]Salaire)
```

Référence

GRAPHE.

25

Images

Formats supportés

Les tableaux suivants décrivent les divers formats d'images supportés par 4e Dimension sous MacOS et Windows, en fonction des opérations réalisées :

- **Copier et Coller**

	PICT	EMF	WMF	BITMAP
MacOS	Oui	-	-	-
Windows	Oui	Oui encapsulé en PicComment	Oui encapsulé en PicComment	Oui converti en PICT Macintosh

- **Affichage**

	PICT	QuickTime	WMF encapsulé	EMF encapsulé
MacOS	Oui	Oui	Non	Non
Windows	Oui	Oui NT & WIN 9x + QT 4	Oui	Oui

A propos des fichiers WMF (Windows MetaFile)
Ces fichiers doivent être des fichiers "positionnables", comportant un en-tête précisant la dimension du dessin et sa résolution. Si cet en-tête est absent, 4D ne pourra pas lire le dessin. Les fichiers WMF sont l'équivalent sous Windows des fichiers PICT, ils peuvent contenir des données vectorielles et bitmap correspondant aux primitives de bases de dessin de chacun des environnements. L'avantage de l'utilisation d'images WMF dans votre base de données est leur plus grande rapidité d'affichage sous Windows (nul besoin de conversion), ainsi que leur universalité. Toutes les applications Windows sont à même d'exporter ce type de fichiers. Attention toutefois, si vous utilisez ce type d'images dans votre base, vous ne pourrez pas les visualiser sous MacOS.

A propos des fichiers EMF (Windows Enhanced Metafile)
Ces fichiers sont une amélioration du format WMF. Les applications Windows doivent progressivement l'adopter. L'avantage de ces fichiers est qu'ils peuvent contenir des primitives de dessin plus élaborées que les fichiers WMF, comme par exemple les Beziers ou les transformations.

Utiliser QuickTime d'Apple avec 4D

4D s'appuie sur QuickTime d'Apple pour gérer la compression et la conversion d'images dans vos bases de données.

• Compression

Apple a ajouté des "opcodes" aux spécifications PICT d'origine. Ainsi, les applications MacOS peuvent manipuler des images QuickTime sans modification. Quand l'application demande au système de dessiner une image contenant des données encapsulées avec QuickTime, l'opcode QuickTime est ignoré si QuickTime n'est pas installé ; si QuickTime est présent, l'image est décompressée et affichée. Cette technologie est transparente pour l'utilisateur et utilise une mémoire minimale, car une image d'un méga octets peut être stockée dans une PICT de 40 kilo-octets, et n'a pas besoin d'être décompressée avant l'affichage.
Sous Windows, 4D requiert que la **version 4 minimum de QuickTime** soit installée pour que vous puissiez utiliser la compression/décompression d'images sur cette plate-forme.

Note : Les commandes utilisant QuickTime mais faisant appel à des fichiers disque (CHARGER ET COMPRESSER IMAGE et COMPRESSER FICHIER IMAGE) ne fonctionnent toutefois pas sous Windows, quelle que soit la version de QuickTime installée.

• Conversion

Des commandes 4D telles que ECRIRE FICHIER IMAGE vous permettent de convertir et de sauvegarder sur disque divers types d'images dans différents formats. La plupart de ces commandes nécessitent la présence de QuickTime. Sous Windows, 4D requiert que la **version 4 minimum de QuickTime** soit installée pour que vous puissiez convertir des images sur cette plate-forme.

• Code de conversion QuickTime 4

Voici la liste des codes de conversion fournis en standard par QuickTime 4. Chaque code est formé impérativement de 4 caractères. A noter que cette liste peut varier en fonction des machines, QuickTime 4 permettant l'ajout de routines de conversion personnalisées. Utilisez la commande LISTE TYPES IMAGES pour obtenir la liste des formats QuickTime disponibles sur une machine.

Codes QuickTime 4	Noms
PICT	QuickDraw PICT (MacOS)
PICS	PICS
GIFf	GIF (Graphic Interchange Format)
PNGf	PNG (Portable Netword Graphic)
TIFF	TIFF (Tagged Image File)
8BPS	Photoshop (2.5 & 3.0)
SGI	Silicon Graphics
BMPf	BMP (Bitmap)

JPEG	JPEG (Joint Photographic Experts Group)
JPEG	JFIF
PNTG	MacPaint
TPIC	TGA (Targa)
qdgx	QuickDraw GX Picture (si QuickDraw GX installé)
qtif	QuickTime Image
FPix	FlashPix

Erreurs de compression ou de conversion d'image

Le code d'erreur -9955 est retourné par 4D quand vous essayez d'utiliser une commande de compression ou de conversion d'image alors que QuickTime n'est pas installé dans votre système. D'autres erreurs générées par QuickTime peuvent être aussi retournées. Vous pouvez intercepter ces erreurs en utilisant une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Opérateurs sur les images

4e Dimension vous permet d'effectuer des **opérations** sur les images 4D, telles que la concaténation, la superposition, etc. Ce point est traité dans la section Opérateurs sur les images.

Référence

BLOB VERS IMAGE, CHARGER ET COMPRESSER IMAGE, COMPRESSER FICHIER IMAGE, COMPRESSER IMAGE, ECRIRE FICHIER IMAGE, ENREGISTRER IMAGE, IMAGE VERS BLOB, LIRE FICHIER IMAGE, LISTE TYPES IMAGES, PROPRIETES IMAGE, Taille image.

COMPRESSER IMAGE

version 5

COMPRESSER IMAGE (image; type; qualité)

Paramètre	Type		Description
image	Image	→	Image à compresser
		←	Image compressée
type	Alpha	→	Type de compression (4 caractères)
qualité	Numérique	→	Qualité de compression (1...1000)

Description

La commande COMPRESSER IMAGE compresse l'image contenue dans un champ ou une variable de type Image.

Le paramètre type est une chaîne de quatre caractères indiquant le type de compresseur.

Le paramètre qualité est un entier entre 1 et 1000 indiquant la qualité de l'image compressée. En général, réduire la qualité permet une meilleure compression de l'image.

Attention : Le ratio de compression possible pour une qualité donnée dépend de la taille de l'image et de l'image que vous voulez compresser. La compression de petites images peut ne pas produire de réduction de taille significative.

Référence

CHARGER ET COMPRESSER IMAGE, COMPRESSER FICHIER IMAGE, Introduction aux images.

CHARGER ET COMPRESSER IMAGE

version 5

CHARGER ET COMPRESSER IMAGE (document; type; qualité; image)

Paramètre	Type		Description
document	DocRef	→	Numéro de référence du document
type	Alpha	→	Type de compression
qualité	Numérique	→	Qualité de compression (1...1000)
image	Image	←	Image compressée

Description

CHARGER ET COMPRESSER IMAGE compresse une image chargée d'un document sur disque.

Vous pouvez ouvrir un document PICT à l'aide de la commande Ouvrir document et utiliser le numéro de référence retourné par cette fonction pour charger et compresser la PICT qu'il contient. Cette commande charge l'image en mémoire, la compresse en utilisant le type et la qualité que vous spécifiez et la retourne dans image.

L'image est chargée en mémoire avant d'être compressée. S'il n'y a pas assez de mémoire pour charger l'image, utilisez COMPRESSER FICHIER IMAGE avant d'appeler CHARGER ET COMPRESSER IMAGE.

Le paramètre type est une chaîne de 4 caractères indiquant le type de compression. Si type est une chaîne vide, image est chargée mais n'est pas compressée.

Le paramètre qualité est un Entier entre 1 et 1000 indiquant la qualité de l'image compressée. En général, réduire la qualité permet une meilleure compression de l'image.

Attention : Le ratio de compression possible pour une qualité donnée dépend de la taille de l'image et de l'image que vous voulez compresser. La compression de petites images peut ne pas produire de réduction de taille significative.

Exemple

L'exemple suivant affiche une boîte de dialogue d'ouverture de fichiers qui vous permet de sélectionner un fichier PICT. L'image est chargée en mémoire, compressée et stockée dans une variable de type Image, puis le fichier est refermé.

```
vRéf:=Ouvrir document ("";"PICT")
Si (OK=1)
⇒    CHARGER ET COMPRESSER IMAGE(vRéf;"jpeg";500;vImage)
     FERMER DOCUMENT(vRéf)
Fin de si
```

Référence

COMPRESSER FICHIER IMAGE, COMPRESSER IMAGE, ENREGISTRER IMAGE, Introduction aux images.

COMPRESSER FICHIER IMAGE (document; type; qualité)

Paramètre	Type		Description
document	DocRef	→	Numéro de référence du document
type	Alpha	→	Type de compression
qualité	Numérique	→	Qualité de compression (1...1000)

Description

COMPRESSER FICHIER IMAGE compresse un document image sur disque. Utilisez cette commande pour compresser une image que vous savez ne pas pouvoir charger dans la mémoire disponible. Une fois compressée, elle peut être chargée en mémoire grâce à CHARGER ET COMPRESSER IMAGE.

Le paramètre type est une chaîne de 4 caractères indiquant le type de compression.

Le paramètre qualité est un entier entre 1 et 1000 indiquant la qualité de l'image compressée. En général, réduire la qualité permet une meilleure compression de l'image.

Attention : Le ratio de compression possible pour une qualité donnée dépend de la taille de l'image et de l'image que vous voulez compresser. La compression de petites images peut ne pas produire de réduction de taille significative.

Exemple

L'exemple suivant affiche une boîte de dialogue d'ouverture de fichiers qui vous permet de sélectionner un fichier PICT. Seuls les fichiers de type PICT seront affichés. L'image est compressée puis chargée en mémoire et stockée dans une variable de type Image. Le fichier est ensuite fermé.

```
    vRéf:=Ouvrir document ("";"PICT")
    Si (OK=1)
⇒       COMPRESSER FICHIER IMAGE(vRef;"jpeg";500)
        CHARGER ET COMPRESSER IMAGE(vRéf;"jpeg";500;Image)
        FERMER DOCUMENT(vRéf)
    Fin de si
```

Référence

CHARGER ET COMPRESSER IMAGE, COMPRESSER IMAGE, ENREGISTRER IMAGE.

ENREGISTRER IMAGE (document; image)

Paramètre	Type		Description
document	DocRef	→	Numéro de référence du document
image	Image	→	Image à enregistrer

Description

ENREGISTRER IMAGE enregistre image dans le document document, créé à l'aide de la fonction Creer document.

Exemple

L'exemple suivant crée un document et y stocke une image :

```
      vRéf:=Creer document("";"PICT")
      Si (OK=1)
⇒        ENREGISTRER IMAGE(vRéf;vPict)
         FERMER DOCUMENT(vRéf)
      Fin de si
```

Référence

CHARGER ET COMPRESSER IMAGE, COMPRESSER FICHIER IMAGE.

IMAGE VERS GIF (imagePICT; blobGIF)

Paramètre	Type		Description
imagePICT	Image	→	Champ ou variable image
blobGIF	BLOB	←	BLOB contenant l'image de type GIF

Description

La commande IMAGE VERS GIF permet de créer une image au format GIF à partir d'une image (de type PICT) stockée dans une variable ou un champ 4D.

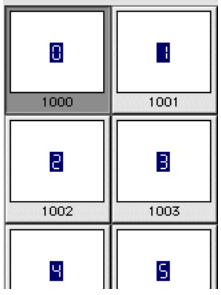
Vous passez dans le paramètre imagePICT une variable ou un champ 4D de type image, et dans le paramètre blobGIF, une variable ou un champ de type BLOB. Après l'exécution de la commande, blobGIF contient l'image au format GIF.

Note : Le format GIF est un format d'image comportant au plus 256 couleurs. Si l'image PICT d'origine en possède davantage, certaines couleurs seront perdues.
Le GIF généré par cette commande est optimisé en fonction des couleurs. Il est de type 87a (opaque) et normal (non entrelacé).

L'image incluse dans blobGIF pourra par la suite être enregistrée dans un fichier à l'aide de la commande BLOB VERS DOCUMENT, ou être utilisée en vue d'une publication sur le Web.
Si la conversion s'est déroulée correctement, la variable système *OK* prend la valeur 1. Sinon, elle prend la valeur 0.

Exemple

Vous souhaitez générer à la volée une image GIF affichant un compteur de connexions. Dans la bibliothèque d'images de la base, placez tous les chiffres sous forme d'images :



Dans la Méthode base Sur connexion Web, vous pouvez écrire :

```
   `Méthode base Sur connexion Web
Si (Contexte Web)
   ...
Sinon
   C_BLOB ($blob)
   Au cas ou
      ...
   : ($1="/4dcgi/counter") `Génération du compteur GIF
         `Lorsque 4D détecte cet URL lors de l'envoi de la page statique
      $blob:=gifcounter (◊nbHits)    `Calcul de l'image gif
         `La variable ◊nbHits contient le nombre de connexions
      ENVOYER BLOB HTML ($blob;"image/gif")
         `Insertion de l'image et envoi au browser
   ...
   Fin de cas
Fin de si
```

Voici la méthode *gifcounter* :

```
   `Méthode projet gifcounter
C_ENTIER LONG($1)
C_IMAGE($img)
C_BLOB($0)
Si ($1=0)
   $ndigits:=1
Sinon
   $ndigits:=1+Longueur(Chaine($1))
Fin de si
Si ($ndigits<5)
   $ndigits:=5
Fin de si
$div:=10^($ndigits-1)
Boucle ($i;1;$ndigits)
   $ref:=Ent($1/$div)%10
   LIRE IMAGE DANS BIBLIOTHEQUE($ref+1000;picture)
   $img:=$img+picture
   $div:=$div/10
Fin de boucle
⇒  IMAGE VERS GIF($img;$0)
```

Lors de l'envoi de la page sur le browser, 4D affiche alors une image GIF du type suivant :

Variables et ensembles système

Si la conversion s'est déroulée correctement, la variable système *OK* prend la valeur 1. Sinon, elle prend la valeur 0.

IMAGE VERS BLOB

version 6.7

IMAGE VERS BLOB (image; blobImage; format)

Paramètre	Type		Description
image	Image	→	Champ ou variable image
blobImage	BLOB	←	BLOB devant contenir l'image convertie
format	Alpha (4)	→	Format d'image (4 caractères)

Description

La commande IMAGE VERS BLOB convertit une image stockée dans une variable ou un champ 4D dans un autre format, et place l'image résultante dans un BLOB.

Vous passez dans le paramètre image une variable ou un champ 4D de type image et dans le paramètre blobImage la variable ou le champ BLOB devant contenir l'image convertie.

Vous passez dans le paramètre format une chaîne de 4 caractères indiquant le format de conversion souhaité.
Ce format peut être :
• soit un code QuickTime (cf. description de la commande LISTE TYPES IMAGES, auquel cas QuickTime version 4 minimum devra être installé sur le poste,
• soit GIFf (format GIF) ou WBMP (Wireless Bitmap), ces deux codes ne nécessitant pas la présence de QuickTime.

Après l'exécution de la commande, blobImage contient l'image au format souhaité.

Si la conversion s'est déroulée correctement, la variable système *OK* prend la valeur 1. Si la conversion échoue (absence de QuickTime 4 ou convertisseur non disponible), *OK* prend la valeur 0 et le BLOB est généré vide (0 octet).

Référence

BLOB VERS IMAGE, ECRIRE FICHIER IMAGE, IMAGE VERS GIF, LISTE TYPES IMAGES.

BLOB VERS IMAGE

version 6.7

BLOB VERS IMAGE (blobImage; image)

Paramètre	Type		Description
blobImage	BLOB	→	BLOB contenant une image
image	Image	←	Champ ou variable image 4D

Description

La commande BLOB VERS IMAGE place dans un champ ou une variable image 4D une image stockée dans un BLOB, quel que soit son format initial (compatible QuickTime 4).

Attention : Cette commande nécessite sous MacOS et Windows la présence de QuickTime version 4 minimum. Si QuickTime 4 n'est pas installé, la commande ne fait rien.

Le fonctionnement de cette commande est analogue à celui de la commande LIRE FICHIER IMAGE ; elle s'applique simplement à un BLOB et non à un fichier. Elle permet d'afficher à tout moment des images stockées en format natif dans des BLOBs à l'aide, par exemple, de la commande DOCUMENT VERS BLOB ou IMAGE VERS BLOB.

Vous passez dans le paramètre blobImage le BLOB contenant l'image. L'image peut être de tout format compatible QuickTime (version 4 minimum). Vous pouvez obtenir la liste des formats disponibles à l'aide de la commande LISTE TYPES IMAGES. Pour une description complète des codes de format standard QuickTime 4, reportez-vous à la description de la commande LISTE TYPES IMAGES.
Vous passez dans le paramètre image la variable ou le champ 4D de type image devant afficher l'image.

Note : Le format interne de l'image est conservé par QuickTime au sein de la variable ou du champ 4D. Par conséquent, il sera nécessaire de disposer de QuickTime pour afficher l'image dans 4D.

Après l'exécution de la commande, image contient l'image affichable dans 4D.

Si la commande a été exécutée correctement, la variable système *OK* prend la valeur 1. En cas d'échec (absence de QuickTime 4, BLOB ne contenant pas d'image, format d'image inconnu...), *OK* prend la valeur 0 et le champ ou la variable image 4D est vide.

Référence

IMAGE VERS BLOB, LIRE FICHIER IMAGE, LISTE TYPES IMAGES.

ECRIRE FICHIER IMAGE

version 6.7

ECRIRE FICHIER IMAGE (nomFichier; image{; format})

Paramètre	Type		Description
nomFichier	Alpha	→	Nom ou chemin d'accès complet du fichier à écrire, ou chaîne vide
image	Image	→	Champ ou variable image à écrire
format	Alpha (4)	→	Code QuickTime de format d'export d'image (4 caractères), Par défaut = PICT

Description

La commande ECRIRE FICHIER IMAGE vous permet de sauvegarder dans un fichier sur disque l'image passée dans le paramètre image, au format défini par format.

Important : Cette commande utilise sous MacOS et Windows les routines de conversion de QuickTime (version 4 minimum recommandée). Si QuickTime n'est pas installé, la commande crée par défaut un fichier au format PICT.

Vous pouvez passer dans nomFichier le chemin d'accès complet du fichier à créer, ou uniquement le nom du fichier — auquel cas le fichier sera créé à côté du fichier de structure de la base. Sous Windows, vous devez également passer l'extension du fichier à créer.
Si vous passez une chaîne vide ("") dans nomFichier, la boîte de dialogue standard d'enregistrement de fichier apparaît, permettant à l'utilisateur de désigner le nom, l'emplacement et le format du fichier à créer.

Passez dans image la variable ou le champ image contenant l'image à stocker sur le disque.

Le paramètre optionnel format vous permet de définir le format dans lequel l'image doit être sauvegardée. Ce paramètre doit comporter 4 caractères, correspondant à un code QuickTime. Vous pouvez obtenir la liste des formats disponibles à l'aide de la commande LISTE TYPES IMAGES.
Si vous ne passez pas le paramètre format ou si QuickTime n'est pas installé, le fichier image est créé au format PICT.

Si l'exécution de la commande est correcte, la variable système *Document* contient le chemin d'accès complet du fichier créé et la variable système *OK* prend la valeur 1. En cas d'échec, *OK* prend la valeur 0.

Référence

IMAGE VERS BLOB, Introduction aux images, LIRE FICHIER IMAGE, LISTE TYPES IMAGES.

LIRE FICHIER IMAGE

Images

version 6.7

LIRE FICHIER IMAGE (nomFichier; image)

Paramètre	Type		Description
nomFichier	Alpha	→	Nom ou chemin d'accès complet du fichier à lire, ou chaîne vide
image	Image	←	Champ ou variable recevant l'image

Description

La commande LIRE FICHIER IMAGE vous permet d'ouvrir l'image stockée dans le fichier disque désigné par nomFichier et de la placer dans le champ ou la variable 4D image.

Important : Cette commande utilise sous MacOS et Windows les routines de conversion de QuickTime (version 4 minimum recommandée). Si QuickTime n'est pas installé, la commande ne peut ouvrir que les fichiers au format PICT.

Vous pouvez passer dans nomFichier le chemin d'accès complet du fichier à lire, ou uniquement le nom du fichier — auquel cas il doit se trouver à côté du fichier de structure de la base. Sous Windows, vous devez également passer l'extension du fichier.
Si vous passez une chaîne vide ("") dans nomFichier, la boîte de dialogue standard d'ouverture de documents apparaît, permettant à l'utilisateur de sélectionner le fichier à lire, ainsi que les formats disponibles (fournis par QuickTime 4).

Vous pouvez obtenir la liste des formats disponibles à l'aide de la commande LISTE TYPES IMAGES.

Passez dans image la variable ou le champ image devant recevoir l'image lue.

Note : Le format interne de l'image est conservé par QuickTime au sein de la variable ou du champ 4D. Par conséquent, il sera nécessaire de disposer de QuickTime pour relire l'image dans 4D.

Si l'exécution de la commande est correcte, la variable système *Document* contient le chemin d'accès complet du fichier ouvert et la variable système *OK* prend la valeur 1. En cas d'échec, *OK* prend la valeur 0.

Référence

BLOB VERS IMAGE, ECRIRE FICHIER IMAGE, Introduction aux images, LISTE TYPES IMAGES.

LISTE TYPES IMAGES (tabFormats{; tabNoms})

Paramètre	Type		Description
tabFormats	Tab Alpha (4)	←	Codes QuickTime des formats d'import/export disponibles
tabNoms	Tab Alpha	←	Noms des formats

Description

La commande LISTE TYPES IMAGES remplit le tableau tabFormats avec les codes QuickTime d'import/export d'images disponibles sur la machine où elle est exécutée.

Le tableau tabNoms, optionnel, permet de récupérer le nom de chaque format d'image. Les noms des formats sont plus explicites que leurs codes.

Bien entendu, cette commande requiert que QuickTime (version 4 minimum) soit installé sur la machine. Dans le cas contraire, tabFormats contient uniquement le format PICT.

LISTE TYPES IMAGES peut être utilisée pour vérifier la présence des formats d'images requis pour des bases. Cette fonction s'avère particulièrement utile lorsqu'une base emploie des formats personnalisés, non installés par défaut (possibilité offerte par QuickTime 4). Les informations recueillies dans le tableau tabNoms permettent en outre de construire et d'afficher un pop up menu contenant les formats d'export d'images disponibles.

Codes de conversion QuickTime 4

Voici la liste des codes de conversion fournis en standard par QuickTime 4. Chaque code est formé impérativement de 4 caractères. A noter que cette liste peut varier en fonction des machines, QuickTime 4 permettant l'ajout de routines de conversion personnalisées.

Codes QuickTime 4	*Noms*
PICT	QuickDraw PICT (MacOS)
PICS	PICS
GIFf	GIF (Graphic Interchange Format)
PNGf	PNG (Portable Netword Graphic)
TIFF	TIFF (Tagged Image File)
8BPS	Photoshop (2.5 et 3.0)
.SGI	Silicon Graphics
BMPf	BMP (Bitmap)
JPEG	JPEG (Joint Photographic Experts Group)
JPEG	JFIF
PNTG	MacPaint
TPIC	TGA (Targa)
qdgx	QuickDraw GX Picture (si QuickDraw GX installé)
qtif	QuickTime Image
FPix	FlashPix

Ces codes QuickTime d'import/export d'images sont utilisés en particulier par les commandes ECRIRE FICHIER IMAGE et LIRE FICHIER IMAGE.

Référence

BLOB VERS IMAGE, ECRIRE FICHIER IMAGE, IMAGE VERS BLOB, LIRE FICHIER IMAGE.

Taille image

version 5

Taille image (image) → Numérique

Paramètre	Type		Description
image	Image	→	Image pour laquelle vous voulez connaître la taille en octets
Résultat	Numérique	←	Taille en octets de l'image

Description

Taille image retourne la taille de l'image image en octets.

Référence

PROPRIETES IMAGE.

PROPRIETES IMAGE (image; largeur; hauteur{; hOffset{; vOffset{; mode}}})

Paramètre	Type		Description
image	Image	→	Image sur laquelle obtenir les informations
largeur	Numérique	←	Largeur de l'image exprimée en pixels
hauteur	Numérique	←	Hauteur de l'image exprimée en pixels
hOffset	Numérique	←	Offset horizontal lorsque l'image est affichée en arrière-plan
vOffset	Numérique	←	Offset vertical lorsque l'image est affichée en arrière-plan
mode	Numérique	←	Mode de transfert lorsque l'image est affichée en arrière-plan

Description

La commande PROPRIETES IMAGE retourne des informations sur l'image que vous avez passée dans le paramètre image.

Les paramètres largeur et hauteur reçoivent la largeur et hauteur réelles de l'image.

Les paramètres hOffset, vOffset et mode reçoivent la position et le mode de transfert de l'image lorsqu'elle est affichée en arrière-plan dans un formulaire.

Référence

Taille image.

CREER IMAGETTE (source; dest{; largeur{; hauteur{; mode{; profondeur}}}})

Paramètre	Type		Description
source	Image	→	Champ ou variable image 4D à passer en imagette
dest	Image	←	Imagette résultante
largeur	Entier	→	Largeur de l'imagette en pixels, Par défaut = 48
hauteur	Entier	→	Hauteur de l'imagette en pixels, Par défaut = 48
mode	Entier	→	Mode de création de l'imagette Par défaut = proportionnelle centrée (6)
profondeur	Entier	→	Nombre de couleurs de l'imagette en bits/pixel Par défaut = profondeur écran courante (0)

Description

La commande CREER IMAGETTE retourne une imagette à partir d'une image source. Les imagettes sont généralement utilisées pour la prévisualisation d'images dans le cadre d'applications multimédia ou de sites Web.

Note : Cette commande ne nécessite pas la présence de QuickTime.

Passez dans source la variable ou le champ image 4D contenant l'image source à réduire sous forme d'imagette, et dans dest la variable ou le champ image 4D devant recevoir l'imagette résultante.

Les paramètres optionnels largeur et hauteur vous permettent de définir la taille en pixels de l'imagette que vous souhaitez obtenir. Si vous omettez ces paramètres, la taille par défaut de l'imagette sera de 48X48 pixels.

Le paramètre optionnel mode vous permet de définir le mode de création de l'imagette, c'est-à-dire la manière dont elle sera réduite. Vous pouvez utiliser l'un des trois modes suivants, accessibles par l'intermédiaire de constantes prédéfinies disponibles dans le thème "Formats d'affichage des images" :

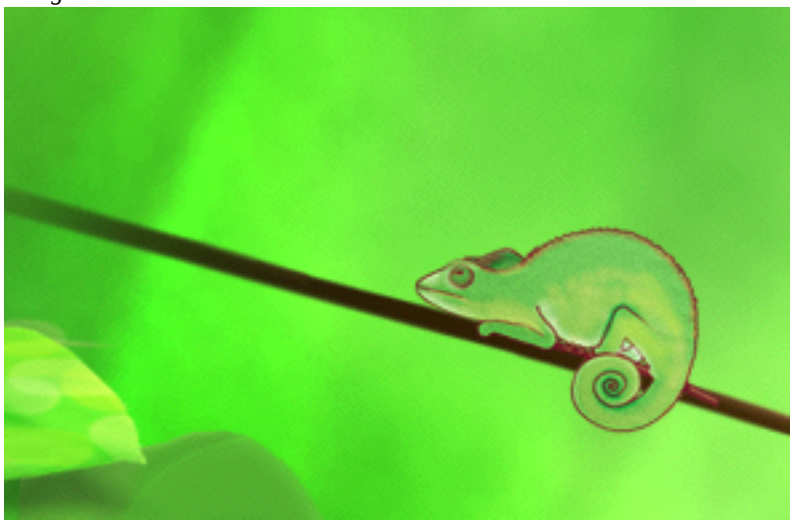
Constante	Type	Valeur
Non tronquée	Entier long	2
Proportionnelle	Entier long	5
Proportionnelle centrée	Entier long	6 (défaut)

Note : Seules ces trois constantes peuvent être utilisées avec CREER IMAGETTE. Les autres constantes du thème "Formats d'affichage des images" ne s'appliquent pas à cette commande.

Si vous omettez le paramètre mode, le mode 6 (Proportionnelle centrée) est appliqué par défaut.

Le résultat des différents modes est illustré ci-dessous :

Image source



Imagettes résultantes (48x48)

• Non tronquée = 2

• Proportionnelle = 5

• Proportionnelle centrée = 6 (mode par défaut)

Note : Avec les modes “Proportionnelle” et “Proportionnelle centrée”, les espaces vides apparaîtront blancs dans les imagettes — lorsque ces modes sont appliqués aux champs ou variables images dans les formulaires 4D, les espaces vides sont transparents.

Le paramètre optionnel profondeur vous permet de définir le nombre de couleurs (c’est-à-dire la profondeur d’écran) à conserver dans l’image de destination. Vous devez passer dans ce paramètre un entier correspondant au nombre de bits par pixel : 1, 2, 4, 8, 16 ou 32. Passez 0 pour utiliser la profondeur écran courante (valeur par défaut).

LISTE IMAGES DANS BIBLIOTHEQUE

version 6.0.2

LISTE IMAGES DANS BIBLIOTHEQUE (RefsImages; NomsImages)

Paramètre	Type		Description
RefsImages	Tableau num	←	Numéros de référence des images stockées dans la bibliothèque d'images
NomsImages	Tableau Alpha	←	Noms des images stockées dans la bibliothèque d'images

Description

La commande LISTE IMAGES DANS BIBLIOTHEQUE retourne les numéros de référence et le nom des images stockées dans la bibliothèque d'images de la base de données.

Après l'appel, vous récupérez les numéros de référence des images dans le tableau RefsImages et leurs noms dans le tableau NomsImages. Les deux tableaux sont synchronisés : le nième élément de RefsImages est le numéro de référence de l'image de la bibliothèque dont le nom est retourné dans le nième élément de NomsImages.

Le tableau RefsImages peut être de type Numérique (Réel), Entier long ou Entier. En mode interprété, si le tableau n'est pas déclaré avant l'appel à LISTE IMAGES DANS BIBLIOTHEQUE, un tableau de type Numérique (Réel) est créé par défaut.

Le tableau NomsImages peut être de type Alpha ou Texte. En mode interprété, si le tableau n'est pas déclaré avant l'appel à LISTE IMAGES DANS BIBLIOTHEQUE, un tableau de type Texte est créé par défaut.

La longueur maximale du nom d'une image de la bibliothèque est de 31 caractères. Si vous utilisez un tableau Alpha pour NomsImages, déclarez-le avec une taille assez grande pour que le nom retourné ne soit pas tronqué.

Si la bibliothèque d'images est vide, les deux tableaux retournés seront vides.

Pour obtenir le nombre d'images contenues dans la bibliothèque, il vous suffit de tester la taille d'un des deux tableaux à l'aide de la fonction Taille tableau.

Exemples

(1) Le code suivant retourne le contenu de la bibliothèque d'images dans les tableaux telRefImage et taNomImage :

```
⇒ LISTE IMAGES DANS BIBLIOTHEQUE(telRefImage;taNomImage)
```

(2) L'exemple suivant teste si la bibliothèque d'images est vide ou non :

```
LISTE IMAGES DANS BIBLIOTHEQUE(telRefImage;taNomImage)
Si (Taille tableau(telRefImage)=0)
   ALERTE("La bibliothèque d'images est vide.")
Sinon
   ALERTE("La bibliothèque d'images contient "+Chaine(Taille tableau(tlRefImage))
                                                              +" images.")
Fin de si
```

(3) L'exemple suivant exporte la bibliothèque d'images vers un fichier stocké sur disque :

```
⇒  LISTE IMAGES DANS BIBLIOTHEQUE($alRefImage;$asPicName)
   $vlNbImages:=Taille tableau($alRefImage)
   Si ($vlNbImages>0)
      REGLER SERIE(12;"")
      Si (OK=1)
         $vsTag:="4DV6PICTURELIBRARYEXPORT"
         ENVOYER VARIABLE($vsTag)
         ENVOYER VARIABLE($vlNbImages)
         gError:=0
         Boucle($vlImage;1;$vlNbImages)
            $vlRefImage:=$alRefImage{$vlImage}
            $vsPicName:=$asPicName{$vlImage}
⇒           LIRE IMAGE DANS BIBLIOTHEQUE(alRefImage{$vlImage};$vgImage)
            Si (OK=1)
               ENVOYER VARIABLE($vlRefImage)
               ENVOYER VARIABLE($vsPicName)
               ENVOYER VARIABLE($vgImage)
            Sinon
               $vlImage:=$vlNbImages+1
               gError:=-108
            Fin de si
         Fin de boucle
         REGLER SERIE(11)
         Si (gError#0)
            ALERTE("La bibliothèque d'images n'a pas pu être exportée, recommencez
                                                    avec davantage de mémoire.")
            SUPPRIMER DOCUMENT (Document)
         Fin de si
      Fin de si
   Sinon
      ALERTE("La bibliothèque d'images est vide.")
   Fin de si
```

Référence

ECRIRE IMAGE DANS BIBLIOTHEQUE, LIRE IMAGE DANS BIBLIOTHEQUE, SUPPRIMER IMAGE DANS BIBLIOTHEQUE.

LIRE IMAGE DANS BIBLIOTHEQUE

Images

version 6.7 (Modifiée)

LIRE IMAGE DANS BIBLIOTHEQUE (refImage | nomImage; image)

Paramètre	Type		Description
refImage | nomImage	Num | Alpha	→	Numéro de référence ou Nom d'une image de la bibliothèque d'images
image	Variable image	←	Image de la bibliothèque d'images

Description

La commande LIRE IMAGE DANS BIBLIOTHEQUE retourne dans image l'image de la bibliothèque dont vous avez passé le numéro de référence dans refImage ou le nom dans nomImage.

Note pour les développeurs de composants : Si vous souhaitez qu'un composant 4D utilise des images stockées dans la Bibliothèque d'images, vous devez utiliser la commande LIRE IMAGE DANS BIBLIOTHEQUE et lui passer un nom d'image comme paramètre. En effet, lorsqu'un composant utilisant des images est installé par 4D Insider, le programme peut renuméroter automatiquement les nouvelles images si des images de même numéro de référence existent déjà dans la base.

S'il n'existe pas d'image de ce numéro ou de ce nom dans la bibliothèque d'images, LIRE IMAGE DANS BIBLIOTHEQUE ne modifie pas le paramètre image.

Exemples

(1) L'exemple suivant retourne dans la variable vgMonImage l'image dont la référence est stockée dans la variable locale $vlRefImage :

```
⇒ LIRE IMAGE DANS BIBLIOTHEQUE($vlRefImage;vgMonImage)
```

(2) L'exemple suivant retourne dans la variable $DDcom_Prot_MonImage l'image nommée "DDcom_Prot_Bouton1" stockée dans la Bibliothèque d'images :

```
⇒ LIRE IMAGE DANS BIBLIOTHEQUE("DDcom_Prot_Bouton1";
                                $DDcom_Prot_MonImage)
```

(3) Reportez-vous au troisième exemple de la commande LISTE IMAGES DANS BIBLIOTHEQUE.

Référence

ECRIRE IMAGE DANS BIBLIOTHEQUE, LISTE IMAGES DANS BIBLIOTHEQUE, SUPPRIMER IMAGE DANS BIBLIOTHEQUE.

Variables et ensembles système

La variable système OK prend la valeur 1 si l’image existe dans la bibliothèque d’images. Sinon, elle prend la valeur zéro.

Gestion des erreurs

S'il n'y a pas assez de mémoire pour retourner l’image, l'erreur –108 est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs.

ECRIRE IMAGE DANS BIBLIOTHEQUE

version 6.0.2

ECRIRE IMAGE DANS BIBLIOTHEQUE (image; refImage; nomImage)

Paramètre	Type		Description
image	Image	→	Nouvelle image
refImage	Numérique	→	Numéro de référence de l'image dans la bibliothèque d'images
nomImage	Alpha	→	Nouveau nom de l'image

Description

La commande ECRIRE IMAGE DANS BIBLIOTHEQUE crée une nouvelle image ou remplace une image existante dans la bibliothèque d'images.

Avant l'appel, vous passez :
• le numéro de référence de l'image dans refImage (compris entre 1 et 32767)
• l'image elle-même dans image.
• Le nom de l'image dans nomImage (longueur maximale : 31 caractères).

S'il existe déjà dans la bibliothèque une image possédant le même numéro de référence, son contenu est remplacé et elle est renommée avec les valeurs que vous avez passées dans image et nomImage.

Si aucune image ne possède le numéro de référence que vous avez passé dans refImage, une nouvelle image est créée dans la bibliothèque d'images.

4D Server : ECRIRE IMAGE DANS BIBLIOTHEQUE ne peut pas être utilisée dans une méthode exécutée sur le poste serveur (procédure stockée ou trigger). Si vous appelez ECRIRE IMAGE DANS BIBLIOTHEQUE sur le serveur, la commande ne fait rien, l'appel est ignoré.

Attention : Les objets de structure (éléments de listes hiérarchiques, lignes de menu, etc.) peuvent se référer à une image de la bibliothèque. Soyez prudent lorsque vous modifiez par programmation une image de la bibliothèque d'images.

Note : Si vous passez une image vide dans image, ou une valeur négative ou nulle dans refImage, la commande ne fait rien.

Exemples

(1) Quel que soit le contenu courant de la bibliothèque d'images, l'exemple suivant ajoute une nouvelle image dans la bibliothèque en cherchant d'abord un numéro de référence d'image unique :

```
⇒    LISTE IMAGES DANS BIBLIOTHEQUE($alRefImage;$asNomImage)
     Repeter
        $vlRefImage:=1+Abs(Hasard)
     Jusque (Chercher dans tableau($alRefImage;$vlRefImage)<0)
⇒    ECRIRE IMAGE DANS BIBLIOTHEQUE(vgImage;$vlRefImage;"Nouvelle Image")
```

(2) L'exemple suivant importe dans la bibliothèque des images stockées dans un document sur disque, créé par le troisième exemple de la commande LISTE IMAGES DANS BIBLIOTHEQUE :

```
     REGLER SERIE(10;"")
     Si (OK=1)
        RECEVOIR VARIABLE($vsTag)
        Si ($vsTag="4DV6BIBLIOTHEQUEIMAGEEXPORT")
           RECEVOIR VARIABLE($vlNbPictures)
           Si ($vlNombreImages)
              Boucle($vlImage;1;$vlNombreImages)
                 RECEVOIR VARIABLE($vlRefImage)
                 Si (OK=1)
                    RECEVOIR VARIABLE($vlNomImage)
                 Fin de si
                 Si (OK=1)
                    RECEVOIR VARIABLE ($vgImage)
                 Fin de si
                 Si (OK=1)
⇒                   ECRIRE IMAGE DANS BIBLIOTHEQUE($vgImage;$vlRefImage;
                                                              $vlNomImage)
                 Sinon
                    $vlImage:=$vlNombreImages+1
                    ALERTE("Ce fichier semble endommagé.")
                 Fin de si
              Fin de boucle
           Sinon
              ALERTE("Ce fichier semble endommagé.")
           Fin de si
        Sinon
           ALERTE("Le fichier ""+Document+"" n'est pas un export de la bibliothèque
                                                              d'images.")
        Fin de si
        REGLER SERIE(11)
     Fin de si
```

Référence

LIRE IMAGE DANS BIBLIOTHEQUE, LISTE IMAGES DANS BIBLIOTHEQUE, SUPPRIMER IMAGE DANS BIBLIOTHEQUE.

Gestion des erreurs

S'il n'y a pas assez de mémoire pour retourner l'image, l'erreur –108 est générée. Notez que des erreurs d'E/S peuvent également être générées (si par exemple le fichier de structure est verrouillé). Vous pouvez intercepter ces erreurs avec une méthode de gestion d'erreurs.

SUPPRIMER IMAGE DANS BIBLIOTHEQUE

Images

version 6.7 (Modifiée)

SUPPRIMER IMAGE DANS BIBLIOTHEQUE (refImage | nomImage)

Paramètre	Type		Description
refImage | nomImage	Num | Alpha	→	Numéro de référence ou Nom d'une image de la bibliothèque d'images

Description

La commande SUPPRIMER IMAGE DANS BIBLIOTHEQUE supprime de la bibliothèque d'images l'image dont vous avez passé le numéro de référence dans refImage ou le nom dans nomImage.
Si ce numéro de référence ou ce nom ne correspond à aucune image, la commande ne fait rien.

4D Server : SUPPRIMER IMAGE DANS BIBLIOTHEQUE ne peut pas être utilisée dans une méthode exécutée sur le poste serveur (procédure stockée ou trigger). Si vous appelez SUPPRIMER IMAGE DANS BIBLIOTHEQUE sur le serveur, il ne se passe rien, l'appel est ignoré.

Attention : Les objets de structure (éléments de liste hiérarchique, lignes de menu, etc.) peuvent se référer à une image de la bibliothèque. Soyez prudent lorsque vous supprimez par programmation une image de la bibliothèque d'images.

Exemples

(1) L'exemple suivant supprime l'image n°4444 de la bibliothèque d'images :

```
⇒ SUPPRIMER IMAGE DANS BIBLIOTHEQUE(4444)
```

(2) L'exemple suivant supprime de la bibliothèque d'images celles dont le nom commence par le symbole dollar ($) :

```
LISTE IMAGES DANS BIBLIOTHEQUE($alRefImage;$asNomImage)
Boucle($vlImage;1;Taille tableau($alRefImage))
   Si ($asNomImage{$vlImage}="$@")
⇒     SUPPRIMER IMAGE DANS BIBLIOTHEQUE($alRefImage{$vlImage})
   Fin de si
Fin de boucle
```

Référence

ECRIRE IMAGE DANS BIBLIOTHEQUE, LIRE IMAGE DANS BIBLIOTHEQUE, LISTE IMAGES DANS BIBLIOTHEQUE.

26

Import-Export

LECTURE ASCII ({table; }document)

Paramètre	Type		Description
table	Table	→	Table dans laquelle effectuer l'import ou Table par défaut si ce paramètre est omis
document	Alpha	→	Document texte à importer

Description

La commande LECTURE ASCII lit les données de document (document texte Windows ou MacOS) et les écrit dans la table table en créant de nouveaux enregistrements.

L'opération d'import s'effectue par le formulaire entrée courant. Les champs et les variables sont lus en fonction du plan où ils se trouvent dans le formulaire entrée. Soyez donc attentif à ce qui se trouve au premier ou au deuxième plan en matière d'objets (champs et variables) dans le formulaire. Le premier objet dans lequel des données sont importées doit être à l'arrière-plan du formulaire. Si le nombre de champs et de variables dans le formulaire ne correspond pas au nombre de champs à importer, les champs supplémentaires sont ignorés. Un formulaire entrée utilisé pour l'import ne peut contenir de boutons. Les objets de sous-formulaire sont ignorés.

Note : Un moyen de s'assurer que les données sont importées dans les bons objets est de sélectionner l'objet dans lequel sera importé le premier champ. Cet objet doit être au premier plan. Pour l'import dans le deuxième champ, placer ce champ devant le premier et ainsi de suite, pour tous les champs et variables. Le champ qui se trouve au premier plan est donc celui dans lequel sont placées les dernières données du document à importer.

L'événement Sur validation est envoyé à la méthode du formulaire pour chaque enregistrement importé. Utilisez cet événement pour copier les données des variables vers les champs, si vous utilisez des variables dans le formulaire d'import.

Le paramètre document peut contenir un chemin d'accès aux noms de volumes et de dossiers. Si vous passez une chaîne vide, la boîte de dialogue standard d'ouverture de fichiers est affichée. Si l'utilisateur annule le dialogue, l'opération d'import est annulée et la variable système OK est mise à 0.

Un thermomètre de progression est affiché pendant l'import. L'utilisateur peut annuler l'opération en cliquant sur le bouton **Stop**. Les enregistrements déjà importés le resteront. Si l'import s'est correctement déroulé, la variable système OK est mise à 1. En cas d'erreur ou d'interruption de l'opération, la variable système OK est mise à 0. Vous pouvez cacher le thermomètre au moyen de la commande SUPPRIMER MESSAGES.

L'opération d'import utilise la table ASCII de la plate-forme sur laquelle est effectué l'import, sauf si vous avez préalablement modifié cette table ASCII avec la commande UTILISER FILTRE. Il est donc possible d'effectuer des imports satisfaisants entre des plates-formes dont les tables ASCII diffèrent.

Lors de l'utilisation de LECTURE ASCII, le délimiteur de champs par défaut est le caractère de tabulation (ASCII 9). Le délimiteur d'enregistrements par défaut est le retour chariot (ASCII 13). Vous pouvez modifier ces valeurs par défaut en assignant de nouvelles valeurs aux variables système FldDelimit et RecDelimit. L'utilisateur peut modifier ces valeurs par défaut dans la boîte de dialogue d'import du mode Utilisation. Comme les champs Texte peuvent contenir des Retours chariot, soyez prudent si vous utilisez le Retour chariot comme délimiteur entre les champs à importer.

Exemple

Cet exemple importe des données d'un document texte. Cette méthode commence par le choix du formulaire. Ici, vous changez les délimiteurs, et vous faites l'import :

```
    FORMULAIRE ENTREE([Personnes]; "Import")
    FldDelimit:=27 ` caractère de délimitation: Escape
    RecDelimit:=10 ` caractère de délimitation: Retour à la ligne
⇒   LECTURE ASCII([Personnes];"Nouvelles Personnes") ` Import du document “Nouvelles
                                                                    Personnes”
```

Référence

ECRITURE ASCII, LECTURE DIF, LECTURE SYLK, UTILISER FILTRE.

Variables et ensembles système

OK prend la valeur 1 si l'import s'est correctement déroulé, sinon elle prend la valeur 0.

ECRITURE ASCII ({table; }document)

Paramètre	Type		Description
table	Table	→	Table depuis laquelle effectuer l'export ou Table par défaut si ce paramètre est omis
document	Alpha	→	Document texte à exporter

Description

La commande ECRITURE ASCII écrit dans document (document texte Windows ou MacOS) les données des enregistrements de la sélection courante de la table du process courant.

L'opération d'export s'effectue par le formulaire sortie courant. Les champs et les variables sont écrits en fonction de l'ordre de saisie dans le formulaire sortie. C'est pourquoi vous devez veiller à utiliser un formulaire sortie ne contenant que les champs ou objets saisissables que vous voulez exporter. Ne placez pas de boutons ou d'autres objets dans le formulaire sortie. Les objets de sous-formulaire sont ignorés.

L'événement Sur chargement est envoyé à la méthode du formulaire pour chaque enregistrement exporté. Utilisez cet événement pour définir les variables utilisées dans le formulaire d'export.

Le paramètre document peut créer un nouveau document ou désigner un document existant. Si le document a le même nom qu'un document existant, ce dernier est remplacé. Le document peut contenir un chemin d'accès de volume ou de dossier. Si vous passez une chaîne vide, la boîte de dialogue standard de sauvegarde de fichiers est affichée. Si l'utilisateur annule ce dialogue, l'opération d'export est annulée et la variable système OK est mise à 0.

Un thermomètre de progression est affiché pendant l'export. L'utilisateur peut annuler l'opération en cliquant sur le bouton **Stop**. Si l'export s'est correctement déroulé, la variable système OK est mise à 1. En cas d'erreur ou d'interruption de l'opération, la variable système OK est mise à 0. Vous pouvez cacher le thermomètre au moyen de la commande SUPPRIMER MESSAGES.

L'opération d'export utilise la table ASCII de la plate-forme de laquelle est effectué l'export, sauf si vous avez préalablement modifié cette table ASCII avec la commande UTILISER FILTRE. On peut ainsi réaliser des exports satisfaisants entre des plates-formes dont les tables ASCII diffèrent.

Lors de l'utilisation de ECRITURE ASCII, le délimiteur de champs par défaut est le caractère de tabulation (ASCII 9). Le délimiteur d'enregistrement est par défaut le retour chariot (ASCII 13). Vous pouvez modifier ces valeurs en assignant de nouvelles valeurs aux deux variables système FldDelimit et RecDelimit. L'utilisateur peut modifier ces valeurs par défaut dans le dialogue d'export du mode Utilisation. Comme les champs Texte peuvent contenir des retours chariot, soyez prudent si vous utilisez le retour chariot comme délimiteur entre les champs à exporter.

Exemple

Cet exemple exporte des données vers un document texte. Cette méthode commence par le choix du formulaire sortie. Ici, vous modifiez les délimiteurs et vous effectuez l'export :

```
    FORMULAIRE SORTIE([Personnes];"Export")
    FldDelimit:=27 ` caractère de délimitation: Escape
    RecDelimit:=10 ` caractère de délimitation: Retour à la ligne
       ` Export vers le document "Nouvelles Personnes"
⇒   ECRITURE ASCII([Personnes];"Nouvelles Personnes")
```

Référence

ECRITURE DIF, ECRITURE SYLK, LECTURE ASCII, UTILISER FILTRE.

Variables et ensembles Système

OK prend la valeur 1 si l'export s'est correctement déroulé, sinon elle prend la valeur 0.

LECTURE SYLK ({table; }document)

Paramètre	Type		Description
table	Table	→	Table dans laquelle effectuer l'import ou Table par défaut si ce paramètre est omis
document	Alpha	→	Document SYLK à importer

Description

La commande LECTURE SYLK lit les données de document (document SYLK Windows ou MacOS) et les écrit dans la table table en créant de nouveaux enregistrements.

L'opération d'import s'effectue par le formulaire entrée courant. Les champs et les variables sont lus en fonction du plan où ils se trouvent dans le formulaire entrée. Soyez donc attentif à ce qui se trouve au premier ou au deuxième plan en matière d'objets (champs et variables) dans le formulaire. Le premier objet dans lequel des données sont importées doit être à l'arrière-plan du formulaire. Si le nombre de champs et de variables dans le formulaire ne correspond pas au nombre de champs à importer, les champs supplémentaires sont ignorés. Un formulaire entrée utilisé pour l'import ne peut contenir de boutons. Les objets de sous-formulaire sont ignorés.

Note : Un moyen de s'assurer que les données sont importées dans les bons objets est de sélectionner l'objet dans lequel sera importé le premier champ. Cet objet doit être au premier plan. Pour l'import dans le deuxième champ, placer ce champ devant le premier et ainsi de suite, pour tous les champs et variables. Le champ qui se trouve au premier plan est donc celui dans lequel sont placées les dernières données du document à importer.

L'événement Sur validation est envoyé à la méthode du formulaire pour chaque enregistrement importé. Utilisez cet événement pour copier les données des variables vers les champs, si vous utilisez des variables dans le formulaire d'import.

Le paramètre document peut contenir un chemin d'accès aux noms de volumes et de dossiers. Si vous passez une chaîne vide, la boîte de dialogue standard d'ouverture de fichiers est affichée. Si l'utilisateur annule le dialogue, l'opération d'import est annulée et la variable système OK est mise à 0.

Un thermomètre de progression est affiché pendant l'import. L'utilisateur peut annuler l'opération en cliquant sur le bouton **Stop**. Les enregistrements déjà importés le resteront. Si l'import s'est correctement déroulé, la variable système OK est mise à 1. En cas d'erreur ou d'interruption de l'opération, la variable système OK est mise à 0. Vous pouvez cacher le thermomètre au moyen de la commande SUPPRIMER MESSAGES.

L'opération d'import utilise la table ASCII de la plate-forme sur laquelle est effectuée l'import, sauf si vous avez préalablement modifié cette table ASCII avec la commande UTILISER FILTRE. Il est donc possible d'effectuer des imports satisfaisants entre des plates-formes dont les tables ASCII diffèrent.

Lors de l'utilisation de LECTURE SYLK, le délimiteur de champs par défaut est le caractère de tabulation (ASCII 9). Le délimiteur d'enregistrements par défaut est le retour chariot (ASCII 13). Vous pouvez modifier ces valeurs par défaut en assignant de nouvelles valeurs aux variables système FldDelimit et RecDelimit. L'utilisateur peut modifier ces valeurs par défaut dans la boîte de dialogue d'import du mode Utilisation. Comme les champs Texte peuvent contenir des Retours chariot, soyez prudent si vous utilisez le Retour chariot comme délimiteur entre les champs à importer.

Exemple

Cet exemple importe des données d'un document SYLK. Cette méthode commence par le choix du formulaire puis déclenche l'import:

```
     FORMULAIRE ENTREE([Personnes]; "Import")
⇒    LECTURE SYLK([Personnes];"Nouvelles Personnes") ` Import du document “Nouvelles
                                                       Personnes”
```

Référence

ECRITURE SYLK, LECTURE ASCII, LECTURE DIF, UTILISER FILTRE.

Variables et ensembles Système

OK prend la valeur 1 si l'import s'est correctement déroulé, sinon elle prend la valeur 0.

ECRITURE SYLK ({table; }document)

Paramètre	Type		Description
table	Table	→	Table de laquelle effectuer l'export ou Table par défaut si ce paramètres est omis
document	Alpha	→	Document SYLK à exporter

Description

La commande ECRITURE SYLK écrit dans document (document SYLK Windows ou MacOS) les données des enregistrements de la sélection courante de la table du process courant.

L'opération d'export s'effectue par le formulaire sortie courant. Les champs et les variables sont écrits en fonction de l'ordre de saisie dans le formulaire sortie. C'est pourquoi vous devez veiller à utiliser un formulaire sortie ne contenant que les champs ou objets saisissables que vous voulez exporter. Ne placez pas de boutons ou autres objets sur le formulaire sortie. Les objets de sous-formulaire sont ignorés.

L'événement Sur chargement est envoyé à la méthode du formulaire pour chaque enregistrement exporté. Utilisez cet événement pour définir les variables utilisées dans le formulaire d'export.

Le paramètre document peut créer un nouveau document ou désigner un document existant. Si le document a le même nom qu'un document existant, ce dernier est remplacé. Le document peut contenir un chemin d'accès de volume ou de dossier. Si vous passez une chaîne vide, la boîte de dialogue standard de sauvegarde de fichiers est affichée. Si l'utilisateur annule ce dialogue, l'opération d'export est annulée et la variable système OK est mise à 0.

Un thermomètre de progression est affiché pendant l'export. L'utilisateur peut annuler l'opération en cliquant sur le bouton **Stop**. Si l'export s'est correctement déroulé, la variable système OK est mise à 1. En cas d'erreur ou d'interruption de l'opération, la variable système OK est mise à 0. Vous pouvez cacher le thermomètre au moyen de la commande SUPPRIMER MESSAGES.

L'opération d'export utilise la table ASCII de la plate-forme de laquelle est effectué l'export, sauf si vous avez préalablement modifié cette table ASCII avec la commande UTILISER FILTRE. On peut ainsi faire des exports satisfaisants entre des plates-formes dont les tables ASCII diffèrent.

Lors de l'utilisation de ECRITURE SYLK, le délimiteur de champs par défaut est le caractère de tabulation (ASCII 9). Le délimiteur d'enregistrement est par défaut le retour chariot (ASCII 13). Vous pouvez modifier ces valeurs en assignant de nouvelles valeurs aux deux variables système FldDelimit et RecDelimit. L'utilisateur peut modifier ces valeurs par défaut dans le dialogue d'export du mode Utilisation. Comme les champs Texte peuvent contenir des retours chariot, soyez prudent si vous utilisez le retour chariot comme délimiteur entre les champs à exporter.

Exemple

Cet exemple exporte des données vers un document SYLK. La méthode commence par le choix du formulaire sortie, puis l'export est déclenché :

```
FORMULAIRE SORTIE([Personnes];"Export")
   ` Export vers le document "Nouvelles Personnes"
⇒  ECRITURE SYLK([Personnes];"Nouvelles Personnes")
```

Référence

ECRITURE ASCII, ECRITURE DIF, LECTURE SYLK, UTILISER FILTRE.

Variables et ensembles Système

OK prend la valeur 1 si l'export s'est correctement déroulé, sinon elle prend la valeur 0.

LECTURE DIF ({table; }document)

Paramètre	Type		Description
table	Table	→	Table dans laquelle effectuer l'import ou Table par défaut si ce paramètre est omis
document	Alpha	→	Document DIF à importer

Description

La commande LECTURE DIF lit les données de document (document DIF Windows ou MacOS) et les écrit dans la table table en créant de nouveaux enregistrements.

L'opération d'import s'effectue par le formulaire entrée courant. Les champs et les variables sont lus en fonction du plan où ils se trouvent dans le formulaire entrée. Soyez donc attentif à ce qui se trouve au premier ou au deuxième plan en matière d'objets (champs et variables) dans le formulaire. Le premier objet dans lequel des données sont importées doit être à l'arrière-plan du formulaire. Si le nombre de champs et de variables dans le formulaire ne correspond pas au nombre de champs à importer, les champs supplémentaires sont ignorés. Un formulaire entrée utilisé pour l'import ne peut contenir de boutons. Les objets de sous-formulaire sont ignorés.

Note : Un moyen de s'assurer que les données sont importées dans les bons objets est de sélectionner l'objet dans lequel sera importé le premier champ. Cet objet doit être au premier plan. Pour l'import dans le deuxième champ, placer ce champ devant le premier et ainsi de suite, pour tous les champs et variables. Le champ qui se trouve au premier plan est donc celui dans lequel sont placées les dernières données du document à importer.

L'événement Sur validation est envoyé à la méthode du formulaire pour chaque enregistrement importé. Utilisez cet événement pour copier les données des variables vers les champs, si vous utilisez des variables dans le formulaire d'import.

Le paramètre document peut contenir un chemin d'accès aux noms de volumes ou de dossiers. Si vous passez une chaîne vide, la boîte de dialogue standard d'ouverture de fichiers est affichée. Si l'utilisateur annule le dialogue, l'opération d'import est annulée et la variable système OK est mise à 0.

Un thermomètre de progression est affiché pendant l'import. L'utilisateur peut annuler l'opération en cliquant sur le bouton **Stop**. Les enregistrements déjà importés le resteront. Si l'import s'est correctement déroulé, la variable système OK est mise à 1. En cas d'erreur ou d'interruption de l'opération, la variable système OK est mise à 0. Vous pouvez cacher le thermomètre au moyen de la commande SUPPRIMER MESSAGES.

L'opération d'import utilise la table ASCII de la plate-forme de laquelle est effectué l'import, sauf si vous avez préalablement modifié cette table ASCII avec la commande UTILISER FILTRE. Il est donc possible d'effectuer des imports satisfaisants entre plates-formes dont les tables ASCII diffèrent.

Lors de l'utilisation de LECTURE DIF, le délimiteur de champs par défaut est le caractère de tabulation (ASCII 9). Le délimiteur d'enregistrements par défaut est le retour chariot (ASCII 13). Vous pouvez modifier ces valeurs par défaut en assignant de nouvelles valeurs aux variables système FldDelimit et RecDelimit. L'utilisateur peut modifier ces valeurs par défaut dans la boîte de dialogue d'import du mode Utilisation. Comme les champs Texte peuvent contenir des Retours chariot, soyez prudent si vous utilisez le Retour chariot comme délimiteur entre les champs à importer.

Exemple

Cet exemple importe des données d'un document DIF. Cette méthode commence par le choix du formulaire, puis effectue l'import :

```
    FORMULAIRE ENTREE([Personnes]; "Import")
       ` Import du document “Nouvelles Personnes”
⇒   LECTURE DIF([Personnes];"Nouvelles Personnes")
```

Référence

ECRITURE DIF, LECTURE ASCII, LECTURE SYLK, UTILISER FILTRE.

Variables et ensembles Système

OK prend la valeur 1 si l'import s'est correctement déroulé, sinon elle prend la valeur 0.

ECRITURE DIF ({table; }document)

Paramètre	Type		Description
table	Table	→	Table de laquelle effectuer l'export ou Table par défaut si ce paramètre est omis
document	Alpha	→	Document DIF à exporter

Description

La commande ECRITURE DIF écrit les données des enregistrements de la sélection courante de la table du process courant dans document (document DIF Windows ou MacOS).

L'opération d'export s'effectue par le formulaire sortie courant. Les champs et les variables sont écrits en fonction de l'ordre de saisie dans le formulaire sortie. C'est pourquoi vous devez veiller à utiliser un formulaire sortie ne contenant que les champs ou objets saisissables que vous voulez exporter. Ne placez pas de boutons ou autres objets sur le formulaire sortie. Les objets de sous-formulaire sont ignorés.

L'événement Sur chargement est envoyé à la méthode du formulaire pour chaque enregistrement exporté. Utilisez cet événement pour définir les variables utilisées dans le formulaire d'export.

Le paramètre document peut créer un nouveau document ou désigner un document existant. Si le document a le même nom qu'un document existant, ce dernier est remplacé. Le document peut contenir un chemin d'accès de volume ou de dossiers. Si vous passez une chaîne vide, la boîte de dialogue standard de sauvegarde de documents s'affiche. Si l'utilisateur annule ce dialogue, l'opération d'export est annulée et la variable système OK est mise à 0.

Un thermomètre de progression est affiché pendant l'export. L'utilisateur peut annuler l'opération en cliquant sur le bouton **Stop**. Si l'export s'est correctement déroulé, la variable système OK est mise à 1. En cas d'erreur ou d'interruption de l'opération, la variable système OK est mise à 0. Vous pouvez cacher le thermomètre au moyen de la commande SUPPRIMER MESSAGES.

L'opération d'export utilise la table ASCII de la plate-forme de laquelle est effectué l'export, sauf si vous avez préalablement modifié cette table ASCII avec la commande UTILISER FILTRE. On peut ainsi faire des exports satisfaisants entre des plates-formes dont les tables ASCII diffèrent.

Lors de l'utilisation de ECRITURE DIF, le délimiteur de champs par défaut est le caractère de tabulation (ASCII 9). Le délimiteur d'enregistrement est par défaut le retour chariot (ASCII 13). Vous pouvez modifier ces valeurs en assignant de nouvelles valeurs aux deux variables système FldDelimit et RecDelimit. L'utilisateur peut modifier ces valeurs par défaut dans le dialogue d'export du mode Utilisation. Comme les champs Texte peuvent contenir des retours chariot, soyez prudent si vous utilisez le retour chariot comme délimiteur entre les champs à exporter.

Exemple

Cet exemple exporte des données vers un document DIF. Cette méthode commence par le choix du formulaire sortie, puis effectue l'export :

```
FORMULAIRE SORTIE([Personnes];"Export")
   ` Export vers le document "Nouvelles Personnes"
⇒  ECRITURE DIF([Personnes];"Nouvelles Personnes")
```

Référence

ECRITURE ASCII, ECRITURE SYLK, LECTURE DIF, UTILISER FILTRE.

Variables et ensembles Système

OK prend la valeur 1 si l'export s'est correctement déroulé, sinon elle prend la valeur 0.

IMPORTER DONNEES

Import-Export

version 6.5

IMPORTER DONNEES (nomFichier{; projet{; *}})

Paramètre	Type		Description
nomFichier	Alpha	→	Chemin d'accès et nom du fichier à importer
projet	BLOB	→	Contenu du projet d'import
		←	Nouveau contenu du projet d'import (si le paramètre * a été passé)
*	*	→	Affichage de la boîte de dialogue d'import et mise à jour du projet

Description

La commande IMPORTER DONNEES permet d'importer des données depuis le fichier nomFichier. 4D peut importer des données au format Texte, Texte de longueur fixe, SYLK, DIF, DBF (dBase), et 4e Dimension.

Si vous passez une chaîne vide dans le nomFichier, IMPORTER DONNEES provoque l'affichage d'une boîte de dialogue standard d'ouverture de fichiers, permettant à l'utilisateur de sélectionner le fichier d'import. Une fois la boîte de dialogue validée, la variable système *Document* contient le chemin d'accès et le nom du fichier d'import. Si l'utilisateur clique sur le bouton **Annuler**, l'exécution est stoppée et la variable système **OK** prend la valeur 0.

• Si vous ne passez pas le paramètre optionnel projet, la boîte de dialogue de paramétrage d'import s'affiche. L'utilisateur peut définir ses paramètres d'import ou charger un projet existant.

Note : Un projet d'import contient tous les paramètres de l'import, tels que les tables et champs d'arrivée, les délimiteurs, etc. Vous définissez ces paramètres dans la boîte de dialogue d'import. Un projet peut être sauvegardé sur disque et chargé. Pour plus d'informations sur cette boîte de dialogue, reportez-vous au manuel *Mode Utilisation*.

• Si vous passez dans le paramètre projet un BLOB contenant un projet d'import valide, l'import s'effectue directement, sans intervention de l'utilisateur.
Dans ce cas, le projet doit avoir été préalablement défini dans la boîte de dialogue de paramétrage d'import, puis conservé. Pour cela, vous disposez de deux solutions :
- le sauvegarder sur disque, puis le charger à l'aide de la commande DOCUMENT VERS BLOB dans le champ ou la variable BLOB que vous passez dans le paramètre projet.
- utiliser la commande IMPORTER DONNEES avec un paramètre projet vide et le paramètre optionnel *, puis stocker le paramètre projet dans un champ BLOB (cf. ci-dessous). Cette solution permet, d'une part, de le conserver avec le fichier de données, et d'autre part d'éviter la phase de chargement dans un BLOB depuis le disque.

Le paramètre optionnel *, s'il est est spécifié, provoque l'affichage de la boîte de dialogue de paramétrage d'import avec les paramétrages définis dans le projet. Ce fonctionnement permet d'utiliser un projet prédéfini, tout en ayant la possibilité de modifier un ou plusieurs paramètres. En outre, le paramètre projet contient, après la fermeture de la boîte de dialogue d'import, les paramètres du "nouveau" projet. Vous pouvez alors le stocker dans un champ BLOB, sur disque, etc.

Si l'import se déroule correctement, la variable système *OK* prend la valeur 1.

Référence

EXPORTER DONNEES, LECTURE ASCII, LECTURE DIF, LECTURE SYLK.

Variables et ensembles système

Si l'utilisateur clique sur **Annuler** dans une des boîtes de dialogue (de sélection de projet ou de paramétrage d'import), la variable système *OK* prend la valeur 0. Si l'import se déroule correctement, la variable système *OK* prend la valeur 1.

EXPORTER DONNEES

Import-Export

version 6.5

EXPORTER DONNEES (nomFichier{; projet{; *}})

Paramètre	Type		Description
nomFichier	Alpha	→	Chemin d'accès et nom du fichier d'export
projet	BLOB	→	Contenu du projet d'export
		←	Nouveau contenu du projet d'export (si le paramètre * a été passé)
*	*	→	Affichage de la boîte de dialogue d'export et mise à jour du projet

Description

La commande EXPORTER DONNEES permet d'exporter des données dans le fichier nomFichier. 4D peut exporter des données au format Texte, Texte de longueur fixe, SYLK, DIF, DBF (dBase), et 4e Dimension.

Si vous passez une chaîne vide dans le nomFichier, EXPORTER DONNEES provoque l'affichage d'une boîte de dialogue standard d'enregistrement de fichiers, permettant à l'utilisateur de définir le nom, le type et l'emplacement du fichier d'export. Une fois la boîte de dialogue validée, la variable système *Document* contient le chemin d'accès et le nom de ce fichier. Si l'utilisateur clique sur le bouton **Annuler**, l'exécution est stoppée et la variable système *OK* prend la valeur 0.

• Si vous ne passez pas le paramètre optionnel projet, la boîte de dialogue de paramétrage d'export s'affiche alors. L'utilisateur peut définir ses paramètres d'export ou charger un projet d'export existant.

Note : Un projet d'export contient tous les paramètres de l'export, tels que les tables et champs exportés, les délimiteurs, etc. Vous définissez ces paramètres dans la boîte de dialogue d'export. Un projet peut être sauvegardé sur disque et chargé. Pour plus d'informations sur cette boîte de dialogue, reportez-vous au manuel *Mode Utilisation*.

• Si vous passez dans le paramètre projet un BLOB contenant un projet d'export valide, l'export s'effectue directement, sans intervention de l'utilisateur.
Dans ce cas, le projet doit avoir été préalablement défini dans la boîte de dialogue de paramétrage d'export, puis conservé. Pour cela, vous disposez de deux solutions :
- le sauvegarder sur disque, puis le charger à l'aide de la commande DOCUMENT VERS BLOB dans le champ ou la variable BLOB que vous passez dans le paramètre projet.
- utiliser la commande EXPORTER DONNEES avec un paramètre projet vide et le paramètre optionnel *, puis stocker le paramètre projet dans un champ BLOB (cf. ci-dessous). Cette solution permet, d'une part, de le conserver avec le fichier de données, et d'autre part d'éviter la phase de chargement dans un BLOB depuis le disque.

Le paramètre optionnel *, s'il est est spécifié, provoque l'affichage de la boîte de dialogue de paramétrage d'export avec les paramétrages définis dans le projet. Ce fonctionnement permet d'utiliser un projet prédéfini, tout en ayant la possibilité de modifier un ou plusieurs paramètres. En outre, le paramètre projet contient, après la fermeture de la boîte de dialogue d'export, les paramètres du "nouveau" projet. Vous pouvez alors le stocker dans un champ BLOB, sur disque, etc.

Si l'export se déroule correctement, la variable système OK prend la valeur 1.

Référence

ECRITURE ASCII, ECRITURE DIF, ECRITURE SYLK, IMPORTER DONNEES.

Variables et ensembles système

Si l'utilisateur clique sur **Annuler** dans une des boîtes de dialogue (d'enregistrement de projet ou de paramétrage d'export), la variable système *OK* prend la valeur 0. Si l'export se déroule correctement, la variable système *OK* prend la valeur 1.

27

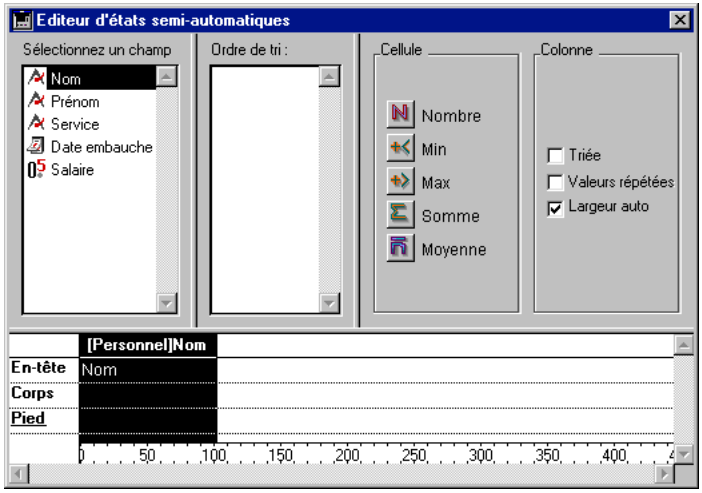
Impressions

ETAT ({table; }document{; *})

Paramètre	Type		Description
table	Table	→	Table à imprimer ou Table par défaut si ce paramètre est omis
document	Alpha	→	Document d'état semi-automatique
*		→	Suppression des boîtes de dialogue d'impression

Description

ETAT imprime un état pour table, à l'aide de l'Editeur d'états semi-automatiques présenté ci-dessous.



Le paramètre document définit un modèle d'état créé dans l'éditeur d'états semi-automatiques et sauvegardé sur disque. Vous sauvegardez un modèle d'état en choisissant **Enregistrer** ou **Enregistrer sous** dans le menu **Fichier** de l'éditeur d'états semi-automatiques. Le document stocke les paramètres de l'état, pas les enregistrements.

Si une chaîne vide ("") est passée dans document, ETAT affiche une boîte de dialogue d'ouverture de fichiers, dans laquelle l'utilisateur peut choisir un modèle d'état à imprimer. Une fois qu'un fichier d'état est sélectionné, les boîtes de dialogue d'impression s'affichent, sauf si le paramètre * a été spécifié — dans ce cas, elles ne s'affichent pas. L'état est alors imprimé.

Si le paramètre document spécifie un document qui n'existe pas (si vous passez, par exemple, Caractere (1) dans document), l'éditeur d'états semi-automatiques s'affiche. Cet éditeur permet à l'utilisateur de construire en totalité son propre état. Pour plus d'informations sur la création d'états à l'aide de l'Editeur d'états semi-automatiques, reportez-vous au manuel *Mode Utilisation* de 4e Dimension.

Lorsque l'Editeur d'états semi-automatiques n'est pas affiché, la variable système OK prend la valeur 1 si un état est imprimé ; sinon elle prend la valeur 0 (zéro) — par exemple si l'utilisateur a cliqué sur **Annuler** dans les boîtes de dialogue d'impression.

Exemples

(1) L'exemple suivant permet à l'utilisateur d'effectuer une recherche dans la table [Personnes], puis imprime automatiquement l'état "Liste détaillée" :

```
      CHERCHER ([Personnes])
      Si (OK=1)
⇒        ETAT ([Personnes]; "Liste détaillée";*)
      Fin de si
```

(2) L'exemple suivant permet à l'utilisateur d'effectuer une recherche dans la table [Personnes], puis de sélectionner le document d'état qui sera ensuite utilisé pour l'impression :

```
      CHERCHER ([Personnes])
      Si (OK=1)
⇒        ETAT ([Personnes]; "")
      Fin de si
```

(3) L'exemple suivant permet à l'utilisateur d'effectuer une recherche dans la table [Personnes], puis affiche l'Editeur d'états semi-automatiques afin que l'utilisateur puisse construire, charger, sauvegarder ou imprimer tout état :

```
      CHERCHER ([Personnes])
      Si (OK=1)
⇒        ETAT ([Personnes]; Caractere(1))
      Fin de si
```

Référence

IMPRIMER ETIQUETTES, IMPRIMER SELECTION.

IMPRIMER ETIQUETTES ({table}{; étiqFichier}{; * | >})

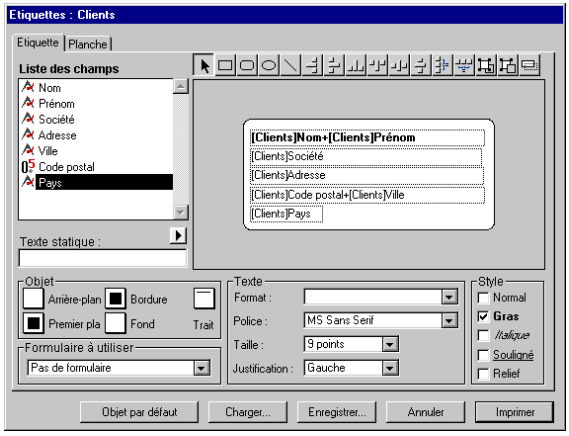
Paramètre	Type		Description
table	Table	→	Table à imprimer ou Table par défaut si ce paramètre est omis
étiqFichier	Alpha	→	Nom de fichier d'étiquettes sur disque
* | >	* | >	→	* pour supprimer les boîtes de dialogue d'impression ou > pour ne pas réinitialiser les paramètres d'impression

Description

IMPRIMER ETIQUETTES vous permet d'imprimer des étiquettes à partir des données de la sélection de table.

Si vous ne spécifiez pas le paramètre étiqFichier, IMPRIMER ETIQUETTES imprime la sélection courante de table sous forme d'étiquettes, en utilisant le formulaire sortie courant du process. Vous ne pouvez pas imprimer de sous-formulaires lorsque vous utilisez cette commande. Pour plus d'informations sur la création d'étiquettes à l'aide de formulaires, reportez-vous au manuel *Mode Structure* de 4e Dimension.

Si vous spécifiez le paramètre étiqFichier, IMPRIMER ETIQUETTES vous permet d'imprimer un document d'étiquettes existant stocké sur disque ou d'ouvrir l'Assistant de création d'étiquettes (affiché ci-dessous).



Par défaut, IMPRIMER ETIQUETTES affiche les boîtes de dialogue d'impression. Si l'utilisateur annule une des deux boîtes de dialogue, l'exécution de la commande est stoppée et l'état n'est pas imprimé.
Vous pouvez supprimer leur affichage en utilisant soit le paramètre optionnel astérisque (*), soit le paramètre optionnel “supérieur à” (>).
• Le paramètre * provoque une impression avec les paramètres d'impression par défaut, ou ceux définis par la commande UTILISER PARAMETRES IMPRESSION.
• Le paramètre > provoque l’impression sans réinitialisation des paramètres d’impression. Ce paramètre est utile lorsque vous souhaitez exécuter successivement plusieurs appels à IMPRIMER ETIQUETTES (par exemple à l’intérieur d’une boucle) tout en conservant des paramètres d’impression personnalisés préalablement définis.
Il est à noter que ces paramètres n’ont pas d’effet si l'assistant de création d'étiquettes est utilisé.

Si l'assistant de création d'étiquettes n'est pas utilisé, la variable système *OK* est mise à 1 si toutes les étiquettes ont été imprimées ; sinon, elle prend la valeur 0 (zéro) (par exemple si l'utilisateur a cliqué sur le bouton **Annuler** dans les boîtes de dialogue d'impression).

Si vous spécifiez le paramètre étiqFichier, les étiquettes sont imprimées avec les paramétrages définis dans étiqFichier. Si étiqFichier est une chaîne vide (""), IMPRIMER ETIQUETTES affiche une boîte de dialogue standard d'ouverture de documents, permettant à l'utilisateur de sélectionner le fichier d'étiquettes à utiliser. Si étiqFichier est le nom d'un fichier qui n'existe pas ou est invalide (si vous passez, par exemple, Caractere(1) dans étiqFichier), l'assistant de création d'étiquettes s'affiche, permettant à l'utilisateur de créer son propre format d'étiquettes.

Exemples

(1) L'exemple suivant imprime des étiquettes à l'aide du formulaire de sortie de la table. L'exemple s'appuie sur deux méthodes. La première est une méthode projet qui désigne le formulaire sortie à utiliser puis imprime les étiquettes :

```
      TOUT SELECTIONNER([Adresses]) ` Sélection de tous les enregistrements
      FORMULAIRE SORTIE ([Adresses]; "ImprimEtiq")` Sélection du formulaire sortie
⇒     IMPRIMER ETIQUETTES([Adresses]) ` Impression des étiquettes
      FORMULAIRE SORTIE ([Adresses];"Sortie")
         ` Rétablissement du formulaire sortie par défaut
```

La seconde méthode est la méthode du formulaire "ImprimEtiq". Le formulaire contient une variable, nommée vEtiq, contenant les champs concaténés. Si le second champ Adresse (Adr2) est vide, il est enlevé par la méthode (notez que cette opération peut être effectuée automatiquement par l'assistant de création d'étiquettes).

La méthode formulaire construit l'étiquette pour chaque enregistrement :

```
   ` Méthode formulaire [Adresses];"Etiquette sortie"
Au cas ou
   : (Evenement formulaire=Sur chargement)
      vEtiq:=[Adresses]Nom1+" "+[Adresses]Nom2+Caractere(13)+[Adresses]Adr1
                                                          +Caractere(13)
      Si ([Adresses]Adr2 # "")
         vEtiq:=vEtiq +[Adresses]Adr2+Caractere(13)
      Fin de si
      vEtiq:=vEtiq+[Adresses]Ville+", "+[Adresses]Département+" "+[Adresses]Code Postal
Fin de cas
```

(2) L'exemple suivant permet à l'utilisateur d'effectuer une recherche sur la table [Employés], et d'imprimer automatiquement les étiquettes “Mes Etiquettes” :

```
   CHERCHER ([Employés])
   Si (OK=1)
⇒     IMPRIMER ETIQUETTES ([Employés];"Mes Etiquettes";*)
   Fin de si
```

(3) L'exemple suivant permet à l'utilisateur d'effectuer une recherche sur la table [Employés], puis de choisir les étiquettes qui doivent être imprimées :

```
   CHERCHER ([Employés])
   Si (OK=1)
⇒     IMPRIMER ETIQUETTES ([Employés];"")
   Fin de si
```

(4) L'exemple suivant permet à l'utilisateur d'effectuer une recherche sur la table [Employés], puis affiche l'assistant de création d'étiquettes afin que l'utilisateur puisse concevoir, sauvegarder, charger et imprimer tout type d'étiquettes :

```
   CHERCHER ([Employés])
   Si (OK=1)
⇒     IMPRIMER ETIQUETTES ([Employés];Caractere(1))
   Fin de si
```

Référence

ETAT, IMPRIMER SELECTION.

IMPRIMER SELECTION ({table}{; }{* | >})

Paramètre	Type		Description
table	Table	→	Table à laquelle appartient la sélection à imprimer ou Table par défaut si ce paramètre est omis
* | >	* | >	→	* pour supprimer les boîtes de dialogue d'impression ou > pour ne pas réinitialiser les paramètres d'impression

Description

La commande IMPRIMER SELECTION imprime la sélection courante de table. Les enregistrements sont imprimés dans le formulaire sortie courant de la table du process en cours. IMPRIMER SELECTION a le même effet que la commande **Imprimer...** du mode Utilisation. Si la sélection courante est vide, IMPRIMER SELECTION ne fait rien.

Par défaut, IMPRIMER SELECTION affiche les boîtes de dialogue d'impression. Si l'utilisateur annule une des deux boîtes de dialogue, l'exécution de la commande est stoppée et l'état n'est pas imprimé.
Vous pouvez supprimer leur affichage en utilisant soit le paramètre optionnel astérisque (*), soit le paramètre optionnel “supérieur à” (>).
• Le paramètre * provoque une impression avec les paramètres d'impression par défaut, ou ceux définis par la commande UTILISER PARAMETRES IMPRESSION.
• Le paramètre > provoque l’impression sans réinitialisation des paramètres d’impression. Ce paramètre est utile lorsque vous souhaitez exécuter successivement plusieurs appels à IMPRIMER SELECTION (par exemple à l'intérieur d’une boucle) tout en conservant des paramètres d’impression personnalisés préalablement définis.

Pendant l'impression, la méthode du formulaire sortie et les méthodes objet du formulaire sont exécutées en fonction des événements sélectionnés dans les fenêtres des Propriétés des formulaires et des objets, en mode Structure, ainsi que des événements effectivement générés :
• Un événement Sur entête est généré juste avant que la zone d'en-tête soit imprimée.
• Un événement Sur impression corps est généré juste avant que l'enregistrement soit imprimé.
• Un événement Sur impression sous total est généré juste avant qu'une zone de rupture soit imprimée.
• Un événement Sur impression pied de page est généré juste avant que la zone de pied de page soit imprimée.

Vous pouvez savoir si IMPRIMER SELECTION est sur le point d'imprimer le premier en-tête en testant Avant selection pendant un événement Sur entête. Vous pouvez également savoir si IMPRIMER SELECTION est sur le point d'imprimer le dernier pied de page, en testant Fin de selection pendant un événement Sur impression pied de page.
Pour plus d'informations, reportez-vous à la description de ces commandes ainsi qu'aux commandes Evenement formulaire et Niveau.

Si IMPRIMER SELECTION est appelé au même moment par deux process différents, l'impression déclenchée par le second process attendra que le premier ait terminé.

Pour imprimer une sélection triée avec des sous-totaux ou des ruptures à l'aide de la commande IMPRIMER SELECTION, vous devez d'abord trier la sélection. Puis vous devez inclure, dans chaque zone de rupture de l'état, une variable associée à une méthode objet assignant le sous-total à la variable. Vous pouvez aussi utiliser des fonction statistiques ou arithmétiques telles que Somme et Moyenne pour assigner des valeurs aux variables. Pour plus d'informations, reportez-vous à la description des commandes Sous total, NIVEAUX DE RUPTURES et CUMULER SUR.

Attention : N'utilisez pas la commande SAUT DE PAGE avec IMPRIMER SELECTION. SAUT DE PAGE est exclusivement réservée à une utilisation combinée avec la commande IMPRIMER LIGNE.

Après un appel à IMPRIMER SELECTION, la variable *OK* prend la valeur 1 si l'impression s'est déroulée correctement. Si l'impression a été interrompue (par exemple l'utilisateur a cliqué sur un bouton Annuler dans les boîtes de dialogue d'impression), la variable OK prend la valeur 0 (zéro).

Exemple

L'exemple suivant sélectionne la totalité des enregistrements de la table [Personnes]. La commande VISUALISER SELECTION est alors appelée pour afficher les enregistrements et permettre à l'utilisateur de sélectionner ceux qu'il souhaite imprimer. Enfin, les enregistrements choisis sont récupérés à l'aide de la commande UTILISER ENSEMBLE et imprimés par IMPRIMER SELECTION :

```
TOUT SELECTIONNER([Personnes]) ` Sélection de tous les enregistrements
VISUALISER SELECTION ([Personnes]; *) ` Affichage des enregistrements
UTILISER ENSEMBLE ("UserSet")
   ` Utiliser uniquement les enregistrements sélectionnés par l'utilisateur
⇒ IMPRIMER SELECTION ([Personnes]) ` Imprimer les enregistrements sélectionnés
```

Référence

CUMULER SUR, Niveau, NIVEAUX DE RUPTURES, Sous total, UTILISER PARAMETRES IMPRESSION.

Page impression → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Numéro de la page en cours d'impression

Description

Page impression retourne le numéro de la page en cours d'impression. Cette fonction vous permet de numéroter automatiquement les pages d'une impression en cours à l'aide de IMPRIMER SELECTION ou du menu Impression dans le mode Utilisation.

Exemple

L'exemple suivant change la position des numéros de page sur un état pour que l'état puisse être reproduit au format recto-verso. Le formulaire pour l'état comporte deux variables qui affichent les numéros de page. Une variable dans le coin bas à gauche (vNumGauche) imprime les numéros de page pairs. Une autre variable dans le coin bas à droite (vNumDroite) imprime les numéros de page impairs. L'exemple teste si le numéro de page est pair ou impair, puis utilise et efface les variables appropriées :

```
      Au cas ou
         : (Evenement formulaire=Sur impression pied de page)
⇒           Si ((Page impression % 2) = 0)  ` Modulo vaut 0 pour un numéro de page pair
                  ` Fixer le numéro de page à gauche
⇒              vNumGauche := Chaine (Page impression)
               vNumDroite := ""  ` Effacer le numéro de page droit
            Sinon  ` Sinon, le numéro de page est impair
               vNumGauche := ""  ` Effacer le numéro de page gauche
⇒              vNumDroite := Chaine (Page impression)  ` Fixer le numéro de page à droite
            Fin de si
      Fin de cas
```

Référence

IMPRIMER SELECTION.

NIVEAUX DE RUPTURES (niveau{; sautPage})

Paramètre	Type		Description
niveau	Numérique	→	Nombre de niveaux de rupture
sautPage	Numérique	→	Niveau de saut de page

Description

NIVEAUX DE RUPTURES spécifie le nombre de niveaux de rupture dans un état créé à l'aide de la commande IMPRIMER SELECTION.

Attention : En mode compilé, vous **devez** appeler NIVEAUX DE RUPTURES et CUMULER SUR avant la génération de chaque état dans lequel vous voulez utiliser des ruptures. Ces commandes activent le traitement des ruptures pour un état. Pour plus d'informations, reportez-vous à la description de la commande Sous total.

Le paramètre niveau indique le dernier niveau de rupture pour lequel vous voulez utiliser des ruptures. Ce nombre doit être inférieur ou égal aux niveaux de tris que vous aurez effectués avant l'impression. Si vous avez effectué un tri sur davantage de niveaux, ces niveaux seront imprimés triés, mais ne comporteront pas de rupture.

Chaque niveau de rupture généré provoquera l'impression de zones de rupture et d'en-tête dans le formulaire. Il doit y avoir au moins autant de zones de rupture dans le formulaire que la valeur que vous avez passée dans niveau. S'il y a davantage de zones de rupture, elles seront ignorées et ne seront pas imprimées.

Le second paramètre (optionnel), sautPage, permet de provoquer un saut de page sur le niveau de rupture de votre choix.

Exemple

L'exemple suivant imprime un état avec deux niveaux de rupture. La sélection est triée sur quatre champs, mais la commande NIVEAUX DE RUPTURES ne spécifie que deux niveaux de rupture. Seul un champ est cumulé à l'aide de la commande CUMULER SUR :

```
      ` Trier sur quatre champs
   TRIER ([Emp]Service; >; [Emp]Titre; >; [Emp]Nom; >; Emp]Prénom; >)
⇒  NIVEAUX DE RUPTURES (2)  ` Fixer 2 niveaux de rupture (Service et Titre)
   CUMULER SUR ([Emp]Salaire)  ` Cumuler sur les salaires
   FORMULAIRE SORTIE ([Emp];"ServiceRessHum")  ` Sélectionner le formulaire à imprimer
   IMPRIMER SELECTION([Emp])  ` Imprimer l'état
```

CUMULER SUR (objet{; objet2; ...; objetN})

Paramètre	Type		Description
objet	Champ | Var	→	Champ ou variable de type numérique à cumuler

Description

CUMULER SUR désigne les champ(s) ou variable(s) à cumuler dans un état créé à l'aide de la commande IMPRIMER SELECTION.

Attention : En mode compilé, vous **devez** appeler NIVEAUX DE RUPTURES et CUMULER SUR avant la génération de chaque état dans lequel vous voulez utiliser des ruptures. Ces commandes activent le traitement des ruptures pour un état. Pour plus d'informations, reportez-vous à la description de la commande Sous total.

Utilisez CUMULER SUR si vous souhaitez calculer des sous-totaux pour des champs ou variables numériques dans un état. CUMULER SUR indique à 4e Dimension qu'il faut stocker les sous-totaux pour chaque élément spécifié dans objet. Les sous-totaux sont cumulés pour chaque niveau de rupture spécifié par la commande NIVEAUX DE RUPTURES.

Exécutez CUMULER SUR avant d'imprimer un état à l'aide de IMPRIMER SELECTION.

Utilisez la fonction Sous total dans la méthode formulaire ou une méthode objet pour retourner le sous-total d'un des objets spécifié dans objet.

Exemples

Reportez-vous à l'exemple de la commande NIVEAUX DE RUPTURES.

Référence

IMPRIMER SELECTION, NIVEAUX DE RUPTURES, Sous total, TRIER.

Sous total (valeurs{; sautPage}) → Numérique

Paramètre	Type		Description
valeurs	Champ	→	Champ ou variable numérique dont vous voulez calculer le sous-total
sautPage	Numérique	→	Niveau de rupture auquel effectuer un saut de page
Résultat	Numérique	←	Sous-total de valeurs

Description

Sous total retourne le sous-total de valeurs pour le niveau de rupture courant ou précédent. Sous total ne fonctionne que dans le cadre d'une sélection triée imprimée par l'intermédiaire de la commande IMPRIMER SELECTION ou de la commande de menu **Imprimer** du mode Utilisation. Le paramètre valeurs doit être de type numérique, entier ou entier long. Vous devez assigner le résultat de la fonction Sous total à une variable placée dans la zone de rupture du formulaire.

Attention : En vue d'une utilisation en mode compilé, vous **devez** utiliser les commandes NIVEAUX DE RUPTURES et CUMULER SUR avant d'imprimer un état sur lequel vous voulez traiter les niveaux de rupture et calculer des sous-totaux. Reportez-vous au paragraphe situé à la fin de cette section.

Sous total doit être appelée dans la méthode d'un formulaire ou d'un objet du formulaire. Avant de lancer une impression, 4e Dimension parcourt les méthodes formulaire et les méthodes objet et, si la fonction Sous total est présente, le traitement des ruptures est déclenché (en mode interprété uniquement).

Le second paramètre (optionnel) de la fonction Sous total est utilisé pour provoquer des sauts de page lors de l'impression. Si sautPage vaut 0, Sous total ne génère aucun saut de page. Si sautPage vaut 1, Sous total génère un saut de page pour chaque niveau de rupture 1. Si sautPage vaut 2, Sous total génère un saut de page pour chaque niveau de rupture 1 et 2, etc.

Si vous souhaitez avoir des sauts de page sur N niveaux de tri, vous devez trier la sélection courante sur N+1 niveaux. Cela vous permet d'effectuer un tri sur un dernier champ, de manière à ce qu'il ne crée pas de rupture indésirable. Si vous souhaitez que le dernier champ trié provoque une rupture, triez-le deux fois.

Conseil : Si vous faites appel à la fonction Sous total dans le formulaire sortie affiché à l'écran, 4e Dimension va afficher un message d'erreur. La fermeture du dialogue d'erreur va provoquer un rafraîchissement de l'écran, donc de nouveau l'exécution de la méthode qui fait appel à Sous total, donc de nouveau un message d'erreur, etc. Pour sortir de ce cercle vicieux, appuyez sur les touche **Alt** + **Maj** (Windows) ou **Option**+**Maj** (Macintosh) et cliquez sur le bouton **Arrêter** dans la fenêtre d'erreur : cela met provisoirement fin aux rafraîchissements d'écran. Choisissez un autre formulaire de sortie pour éviter que le problème ne se répète. Passez en mode Structure pour isoler l'appel à la fonction Sous total par un test (Evenement formulaire = Sur impression sous total) si vous avez l'intention d'utiliser le même formulaire de sortie pour l'écran et l'imprimante.

Exemple

L'exemple suivant est la méthode objet d'une variable intitulée vSalaire, placée dans une zone de rupture d'un formulaire (R0, la zone située au-dessus du marqueur R0). La variable prend la valeur du sous-total du champ Salaire pour ce niveau de rupture :

```
      Au cas ou
      : (Evenement formulaire = Sur impression sous total)
⇒        vSalaire := Sous total ([Employés]Salaire)
      Fin de cas
```

Reportez-vous au chapitre "Les formulaires de sortie et les états" du manuel *Mode Structure* pour plus d'informations sur la construction de formulaires avec des niveaux de ruptures.

Traitement de niveaux de rupture dans les formulaires d'état

Le traitement des niveaux de rupture peut être provoqué de deux manières differentes :
• soit en appelant la fonction Sous total,
• soit en appelant les commandes CUMULER SUR et NIVEAUX DE RUPTURES.
Ces deux techniques permettent d'obtenir le même résultat mais ont chacune leurs avantages.

Appel de la fonction Sous total (seulement en mode interpreté)

Pour provoquer le traitement des niveaux de rupture avec la fonction Sous total, celle-ci doit apparaître dans la méthode formulaire ou dans la méthode d'un objet du formulaire. Avant d'imprimer l'état, 4e Dimension cherche cette fonction dans toutes les méthodes du formulaire.

S'il la trouve, 4e Dimension déclenche le traitement des niveaux de rupture. La fonction n'a pas besoin d'être formellement appelée. Elle peut très bien se trouver dans la méthode d'un objet situé en-dessous la zone de pied de page (et n'est donc jamais imprimé ni exécuté).

Lorsque vous utilisez la fonction Sous total pour activer les ruptures, il est nécessaire de trier sur un niveau de plus que le nombre de niveaux de rupture que vous désirez traiter : si vous voulez traiter deux niveaux de ruptures, il est nécessaire de trier sur trois niveaux.

Appel des commandes CUMULER SUR et NIVEAUX DE RUPTURE

Vous pouvez utiliser les commandes CUMULER SUR et NIVEAUX DE RUPTURES pour provoquer le traitement des niveaux de ruptures. Il faut pour cela que ces deux commandes soient appelées avant l'impression du formulaire. L'appel à la fonction Sous total est encore nécessaire pour afficher les calculs de niveaux intermédiaires. L'utilisation de cette technique permet d'éviter de trier sur un niveau de plus. Il est toutefois obligatoire de trier sur au moins le nombre de niveaux de ruptures désiré.

Comparaisons des deux techniques

Le principal avantage la technique utilisant Sous total est d'être plus pratique, particulièrement en mode Utilisation : il n'est pas nécessaire d'exécuter une méthode avant d'imprimer un état.

Les étapes à suivre pour imprimer un état sont les suivantes :
1. Sélectionner les enregistrements à imprimer.
2. Trier les enregistrements, sur un niveau de plus que le nombre de niveaux de rupture.
3. Imprimer avec la commande **Imprimer** du menu **Fichier**.

4e Dimension inspecte toutes les méthodes du formulaire, rencontre la fonction Sous total, déclenche le traitement des niveaux de rupture et imprime l'état. Cette technique comporte cependant deux inconvénients :
• la fonction Sous total n'a pas l'effet voulu en mode compilé,
• il est nécessaire de trier sur un niveau de plus, ce qui prend du temps dans le cas de sélections importantes.

Il est recommandé d'utiliser les commandes CUMULER SUR et NIVEAUX DE RUPTURES lorsque l'on a l'intention d'utiliser une méthode pour imprimer un état. Les étapes à suivre sont les suivantes :
1. Sélectionner les enregistrements à imprimer,
2. Trier les enregistrements sur autant de niveaux que de niveaux de ruptures,
3. Appeler les commandes CUMULER SUR et NIVEAUX DE RUPTURES,
4. Imprimer l'état avec la commande IMPRIMER SELECTION.

Vous devez appeler les commandes CUMULER SUR et NIVEAUX DE RUPTURES pour déclencher le traitement des niveaux de rupture en mode compilé. Toutefois, seule la commande Sous total permet d'afficher des calculs de sous-totaux dans des formulaires.

Référence

CUMULER SUR, IMPRIMER SELECTION, Niveau, NIVEAUX DE RUPTURES.

Niveau → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Niveau de rupture ou d'en-tête courant

Description

La fonction Niveau sert à déterminer le niveau de rupture ou d'en-tête courant. Elle retourne le numéro du niveau de rupture pendant les événements Sur entête et Sur impression sous total.

Le niveau 0 est le dernier niveau à être imprimé et convient à l'impression d'un total général. Niveau retourne 1 lorsque 4e Dimension imprime une rupture sur le premier champ trié, 2 lorsque 4e Dimension imprime une rupture sur le deuxième champ trié, et ainsi de suite.

Exemple

Cet exemple est une maquette de méthode formulaire. Il traite chaque événement possible lorsqu'un état est imprimé dans un formulaire sortie. Niveau est appelé lorsqu'un en-tête ou une rupture est imprimé(e) :

```
      ` Méthode formulaire pour un formulaire sortie utilisé pour un état
   $vpFormTable:=Table du formulaire courant
   Au cas ou
      ` ...
⇒     : (Evenement formulaire=Sur entête)
         ` La zone en-tête va être imprimée
         Au cas ou
            : (Avant selection($vpFormTable->))
               ` Le code pour la première rupture d'en-tête doit être placé ici
            : (Niveau = 1)
               ` Le code pour la rupture d'en-tête niveau 1 doit être placé ici
            : (Niveau = 2)
               ` Le code pour la rupture d'en-tête niveau 2 doit être placé ici
               ` ...
         Fin de cas
```

```
⇒       : (Evenement formulaire=Sur impression corps)
              ` Un enregistrement va être imprimé
              ` Le code pour chaque enregistrement doit être placé ici
⇒       : (Evenement formulaire=Sur impression sous total)
              ` Une rupture va être imprimée
           Au cas ou
              : (Niveau = 0)
                 ` Le code pour la rupture 0 doit être placé ici
              : (Niveau = 1)
                 ` Le code pour la rupture 1 doit être placé ici
                 ` ...
           Fin de cas
⇒       : (Evenement formulaire=Sur impression pied de page)
           Si (Fin de selection($vpFormTable->))
              ` Le code pour le dernier pied de page doit être placé ici
           Sinon
              ` Le code pour le pied de page doit être placé ici
           Fin de si
     Fin de cas
```

Référence

CUMULER SUR, Evenement formulaire, IMPRIMER SELECTION, NIVEAUX DE RUPTURES.

IMPRIMER ENREGISTREMENT ({table}{; }{* | >})

Paramètre	Type		Description
table	Table	→	Table de laquelle imprimer l'enregistrement courant ou Table par défaut si ce paramètre est omis
* | >	* | >	→	* pour supprimer les boîtes de dialogue d'impression ou > pour ne pas réinitialiser les paramètres d'impression

Description

Cette commande provoque l'impression de l'enregistrement courant de table, sans modifier la sélection courante. Le formulaire sortie courant est utilisé pour l'impression. S'il n'y a pas d'enregistrement courant dans table, IMPRIMER ENREGISTREMENT ne fait rien.

IMPRIMER ENREGISTREMENT permet d'imprimer des sous-formulaires et des objets externes, ce qui n'est pas possible avec la commande IMPRIMER LIGNE.

Note : Si l'enregistrement a subi des modifications qui n'ont pas encore été sauvegardées sur disque, la commande imprime les valeurs les plus récentes, et non celles stockées sur le disque.

Par défaut, IMPRIMER ENREGISTREMENT affiche les boîtes de dialogue d'impression. Si l'utilisateur annule une des deux boîtes de dialogue, l'exécution de la commande est stoppée.
Vous pouvez supprimer leur affichage en utilisant soit le paramètre optionnel astérisque (*), soit le paramètre optionnel "supérieur à" (>).
• Le paramètre * provoque une impression avec les paramètres d'impression par défaut, ou ceux définis par la commande UTILISER PARAMETRES IMPRESSION.
• Le paramètre > provoque l'impression sans réinitialisation des paramètres d'impression. Ce paramètre est utile lorsque vous souhaitez exécuter successivement plusieurs appels à IMPRIMER ENREGISTREMENT (par exemple à l'intérieur d'une boucle) tout en conservant des paramètres d'impression personnalisés préalablement définis.

Exemples

(1) L'exemple suivant imprime l'enregistrement courant de la table [Factures]. Cette méthode est celle d'un bouton **Imprimer** placé dans le formulaire entrée. Lorsque l'utilisateur clique sur ce bouton, l'enregistrement est imprimé dans un formulaire spécialement créé dans ce but.

```
FORMULAIRE SORTIE([Factures];"ImpressionEnregistrement")
   `Sélection du formulaire pour l'impression
⇒  IMPRIMER ENREGISTREMENT([Factures];*)
   `Imprimer les factures (sans dialogues d'impression)
FORMULAIRE SORTIE([Factures];"FormListe")
   `Restauration du formulaire sortie courant
```

(2) L'exemple suivant imprime le même enregistrement courant dans deux formulaires différents. Cette méthode est celle d'un bouton **Imprimer** placé dans un formulaire entrée. Vous souhaitez définir des paramètres d'impression personnalisés et les utiliser pour les deux formulaires.

```
PARAMETRES IMPRESSION  `L'utilisateur définit ses paramètres d'impression
Si (OK=1)
   FORMULAIRE SORTIE([Employés];"Détaillé")
      `Utiliser un premier formulaire d'impression
⇒     IMPRIMER ENREGISTREMENT([Employés];>)
      `Imprimer en utilisant les paramètres définis par l'utilisateur
   FORMULAIRE SORTIE([Employés];"Simplifié")
      `Utiliser un second formulaire d'impression
⇒     IMPRIMER ENREGISTREMENT([Employés];>)
      `Imprimer en utilisant les paramètres définis par l'utilisateur
   FORMULAIRE SORTIE([Employés];"Sortie")
      `Rétablir le formulaire sortie par défaut
Fin de si
```

Référence

IMPRIMER LIGNE.

UTILISER PARAMETRES IMPRESSION ({table; }formulaire)

Paramètre	Type		Description
table	Table	→	Table à laquelle appartient le formulaire ou Table par défaut si ce paramètre est omis
formulaire	Alpha	→	Formulaire à utiliser pour définir les paramètres d'impression

Description

UTILISER PARAMETRES IMPRESSION spécifie qu'une impression doit utiliser les paramètres mémorisés avec formulaire. Les paramètres d'impression sont stockés avec le formulaire au moment où il est sauvegardé en mode Structure.

Dans les trois cas suivants :

• un appel à IMPRIMER SELECTION auquel vous passez le paramètre optionnel *,
• un appel à IMPRIMER ENREGISTREMENT auquel vous passez le paramètre optionnel *,
• une série d'appels à IMPRIMER LIGNE non précédée d'un appel à PARAMETRES IMPRESSION,

... les boîtes de dialogue d'impression ne sont pas affichées et l'impression est effectuée avec les paramètres par défaut. Appeler UTILISER PARAMETRES IMPRESSION vous permet de ne pas afficher les boîtes de dialogue d'impression ET d'utiliser des paramètres d'impression qui ne sont pas ceux par défaut.

Exemple

Plusieurs formulaires (vides) sont créés pour une table nommée [Dessins]. Le format de page du formulaire “PS100” est fixé à 100%, Le format de page du formulaire “PS90” est fixé à 90%, etc. La méthode projet suivante vous permet d'imprimer la sélection d'une table à différentes échelles sans devoir à chaque fois spécifier manuellement l'échelle dans les boîtes de dialogue d'impression (qui ne sont d'ailleurs pas affichées) :

```
      ` Méthode projet IMPRESSION ECHELLE AUTO
      ` IMPRESSION ECHELLE AUTO ( Pointeur ; Chaine {; Entier long } )
      ` IMPRESSION ECHELLE AUTO ( ->[Table]; "Form Sortie" {; Echelle } )
   Si (Nombre de parametres>=3)
⇒     UTILISER PARAMETRES IMPRESSION([Dessins];"PS"+Chaine($3))
      Si (Nombre de parametres>=2)
         FORMULAIRE SORTIE($1->;$2)
      Fin de si
   Fin de si
   Si (Nombre de parametres>=1)
      IMPRIMER SELECTION($1->;*)
   Sinon
      IMPRIMER SELECTION(*)
   Fin de si
```

Une fois que cette méthode projet est écrite, vous pouvez l'appeler ainsi :

```
      ` Recherche des factures courantes
   CHERCHER ([Factures];[Factures]Payé=Faux)
      ` Impression d'un état réduit à 90%
   IMPRESSION ECHELLE AUTO (->[Factures];"Etat Résumé";90)
      ` Impression d'un état réduit à 50%
   IMPRESSION ECHELLE AUTO (->[Factures];"Etat détaillé";50)
```

Référence

IMPRIMER ENREGISTREMENT, IMPRIMER LIGNE, IMPRIMER SELECTION.

PARAMETRES IMPRESSION

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

PARAMETRES IMPRESSION affiche les boîtes de dialogue d'impression. La boîte de dialogue de paramétrage de l'impression s'affiche en premier lieu, suivie de celle d'impression proprement dite.

Vous devez appeler PARAMETRES IMPRESSION avant toute série de commandes IMPRIMER LIGNE. En revanche, PARAMETRES IMPRESSION ne fait rien avec les commandes IMPRIMER SELECTION ou IMPRIMER ETIQUETTES.

La boîte de dialogue d'impression comporte l'option **Imprimer à l'écran** permettant à l'utilisateur de visualiser son impression à l'écran. Vous pouvez présélectionner ou désélectionner cette option par un appel préalable à la commande IMPRESSION ECRAN.

Exemple

Reportez-vous à l'exemple de la commande IMPRIMER LIGNE.

Variables et ensembles système

Si l'utilisateur clique sur les boutons OK dans les deux boîtes de dialogue, la variable système OK prend la valeur 1. Sinon, elle prend la valeur 0.

Référence

IMPRESSION ECRAN, IMPRIMER LIGNE, SAUT DE PAGE.

IMPRESSION ECRAN (écran)

Paramètre	Type		Description
écran	Booléen	→	Impression à l'écran (Vrai) ou non (Faux)

Description

La commande IMPRESSION ECRAN vous permet de sélectionner ou de désélectionner l'option **A l'écran** dans la boîte de dialogue standard d'impression. Si vous passez Vrai dans écran, l'option "à l'écran" sera cochée. Si vous passez Faux, elle ne sera pas cochée. Ce paramétrage est local au process et n'affecte pas les autres process ou utilisateurs.

Exemples

L'exemple suivant sélectionne l'option **A l'écran** pour afficher le résultat d'une recherche de clients à l'écran, puis la désélectionne.

```
     CHERCHER([Clients])
     Si (OK=1)
⇒       IMPRESSION ECRAN (Vrai)
        IMPRIMER SELECTION ([Clients] ; *)
⇒       IMPRESSION ECRAN (Faux)
     Fin de si
```

Référence

IMPRIMER ENREGISTREMENT, IMPRIMER SELECTION, PARAMETRES IMPRESSION.

IMPRIMER LIGNE ({table; }formulaire)

Paramètre	Type		Description
table	Table	→	Table à imprimer, ou Table par défaut si ce paramètre est omis
formulaire	Alpha	→	Formulaire à imprimer

Description

La commande IMPRIMER LIGNE imprime simplement formulaire avec les valeurs courantes des champs et des variables de table. Seule la zone Corps du formulaire (c'est-à-dire tous les éléments compris entre la ligne En-tête et la ligne Corps) est imprimée. Cette commande est généralement utilisée pour imprimer des états particulièrement complexes nécessitant un contrôle total du processus d'impression. IMPRIMER LIGNE ne gère pas les traitements d'enregistrements, ni les ruptures, sauts de pages, en-têtes ou pieds de pages. Vous devez vous-même prendre en charge ces opérations. IMPRIMER LIGNE imprime uniquement des champs et des variables sous une forme fixe.

Comme la commande IMPRIMER LIGNE ne génère pas de saut de page après avoir imprimé un formulaire, elle vous permet de combiner facilement différents formulaires sur la même page. Ainsi, IMPRIMER LIGNE est idéale pour effectuer des impressions complexes impliquant plusieurs tables et plusieurs formulaires. Pour "forcer" 4D à effectuer un saut de page entre deux formulaires, utilisez la commande SAUT DE PAGE.

Les boîtes de dialogue standard d'impression n'apparaissent pas lorsque vous utilisez la commande IMPRIMER LIGNE. L'état généré ne tient pas compte des paramètres d'impression définis en mode Structure pour le formulaire. Il y a deux manières de définir les paramètres d'impression avant d'effectuer une série d'appels à IMPRIMER LIGNE :
• Vous appelez PARAMETRES IMPRESSION. Dans ce cas, vous laissez l'utilisateur définir ses paramètres dans les boîtes de dialogue d'impression.
• Vous appelez UTILISER PARAMETRES IMPRESSION. Dans ce cas, les paramètres sont définis par programmation.

IMPRIMER LIGNE construit chaque page à imprimer en mémoire. La page n'est imprimée que lorsqu'elle est entièrement remplie ou lorsque vous appelez SAUT DE PAGE. Pour vous assurer que la dernière page d'une impression exécutée par l'intermédiaire d'IMPRIMER LIGNE soit effectivement imprimée, vous devez conclure par un appel à la commande SAUT DE PAGE. Sinon, la dernière page, si elle n'est pas pleine, reste en mémoire et n'est pas imprimée.

Les sous-formulaires et les zones externes ne sont pas imprimés avec IMPRIMER LIGNE.

Attention : IMPRIMER LIGNE n'imprime pas les sous-formulaires ni les objets externes. Si vous voulez imprimer uniquement un formulaire comportant de tels objets, utilisez plutôt IMPRIMER ENREGISTREMENT.

IMPRIMER LIGNE ne génère qu'un événement Sur impression corps pour la méthode formulaire.

Exemple

L'exemple suivant produit le même résultat que la commande IMPRIMER SELECTION. Cependant, l'état utilise deux formulaires différents en fonction du type d'enregistrement (chèque émis ou dépôt) :

```
CHERCHER([Opérations]) ` Permettre à l'utilisateur de sélectionner les enregistrements
Si (OK=1)
   TRIER([Opérations]) ` Permettre à l'utilisateur de trier les enregistrements
      Si (OK=1)
            ` Afficher les boîtes de dialogue d'impression
         PARAMETRES IMPRESSION([Opérations])
         Si (OK=1)
            Boucle ($i; 1; Enregistrements trouves([Opérations]))
               Si ([Opérations]Type = "Chèque") ` Si c'est un chèque...
                     ` Utiliser un formulaire de chèque
⇒                 IMPRIMER LIGNE([Opérations]; "SortieChèque")
               Sinon ` Sinon c'est un dépôt donc...
                     ` Utiliser un formulaire de dépôt...
⇒                 IMPRIMER LIGNE ([Opérations]; "SortieDépôt")
               Fin de si
               ENREGISTREMENT SUIVANT([Opérations])
            Fin de boucle
            SAUT DE PAGE `S'assurer que la dernière page est imprimée
         Fin de si
      Fin de si
   Fin de si
```

Référence

PARAMETRES IMPRESSION, SAUT DE PAGE, UTILISER PARAMETRES IMPRESSION.

SAUT DE PAGE {(* | >)}

Paramètre	Type		Description
* | >		→	* Annule l'impression lancée par IMPRIMER LIGNE ou > Rend l'impression prioritaire

Description

La commande SAUT DE PAGE provoque l'impression des données envoyées à l'imprimante et provoque un saut de page. SAUT DE PAGE s'utilise conjointement avec IMPRIMER LIGNE pour forcer des sauts de page et imprimer la dernière page créée en mémoire. N'appelez pas SAUT DE PAGE avec la commande IMPRIMER SELECTION : dans ce cas, il est préférable d'utiliser les routines Sous total ou NIVEAUX DE RUPTURES avec leur paramètre optionnel pour générer des sauts de pages.

Les paramètres * et > sont optionnels.

Le paramètre * vous permet d'annuler une impression lancée avec la commande IMPRIMER LIGNE. L'exécution de cette instruction stoppe immédiatement l'impression en cours.

Le paramètre > modifie le fonctionnement de la commande SAUT DE PAGE. Cette syntaxe provoque deux effets :

• Elle reporte l'impression jusqu'à ce que l'instruction SAUT DE PAGE sans paramètre soit de nouveau exécutée.

• Elle rend l'impression prioritaire. Aucune autre impression ne pourra s'intercaler tant que celle en cours ne sera pas terminée.
Cette seconde option est particulièrement intéressante lorsqu'elle est utilisée dans le cadre d'impressions en files d'attente. Le paramètre > garantit que l'impression sera réalisée à partir d'un seul fichier. Cela permet de réduire le temps d'impression.

Exemples

Reportez-vous à l'exemple de la commande IMPRIMER LIGNE.

Référence

IMPRIMER LIGNE.

28

Interface utilisateur

BEEP

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

La commande BEEP provoque l'émission d'un bip sonore. Votre PC ou votre Macintosh peut émettre un autre son qu'un bip en fonction du son sélectionné dans le tableau de bord de contrôle du son.

ATTENTION : N'appelez pas la commande BEEP à partir d'un process de connexion Web car le bip sonore se produira sur le poste serveur Web 4e Dimension et non sur le poste du browser Web.

Exemple

Dans l'exemple suivant, un bip est émis et une alerte affichée lorsqu'aucun enregistrement n'est trouvé par une recherche :

```
      CHERCHER([Clients];[Clients]Nom=$vsNomAChercher)
      Si (Enregistrements trouves([Clients])=0)
⇒        BEEP
         ALERTE("Il n'y a aucun client de ce nom.")
      Fin de si
```

Référence

JOUER SON.

JOUER SON (nomObjet{; canal})

Paramètre	Type		Description
nomObjet	Alpha	→	Nom de son Windows : extension de fichier .WAV, .MID ou .AVI Toute plate-forme : ressource MacOS 'snd' ou chaîne vide pour stopper un son asynchrone
canal	Numérique	→	Si passé : canal de sortie et exécution asynchrone Si omis : exécution synchrone

Description

Sous Windows, la commande JOUER SON permet de jouer des fichiers Windows de sons (fichiers .WAV), MIDI (fichiers .MID) ou vidéo (fichiers .AVI). Vous passez le chemin d'accès complet du fichier que vous voulez jouer dans nomObjet.

Note : Vous ne pouvez pas jouer des fichiers ou objets multimédia en mode asynchrone. Pour cela, utilisez les services OLE.

Sous MacOS (ou sous Windows dans certaines conditions, cf. paragraphe ci-dessous), JOUER SON joue la ressource son nomObjet.

Le paramètre canal spécifie le canal de sortie de synthétiseur Macintosh. Si le canal n'est pas spécifié, le canal est utilisé pour des sons digitalisés simples et est synchrone. Synchrone signifie que tous les traitements s'arrêtent jusqu'à ce que le son soit entièrement joué. Si canal est égal à 0, le canal est utilisé pour des sons digitalisés simples et est asynchrone. Asynchrone signifie que le traitement ne s'arrête pas et que le son est joué en tâche de fond.

Pour stopper un son synchrone, il faut exécuter l'instruction suivante :

⇒ **JOUER SON** ("";0)

Si votre base fonctionne à la fois sur Macintosh et sur PC, vous pouvez jouer des sons Macintosh sous Windows. Pour cela :
• Sous MacOS, à l'aide d'un éditeur de ressources tel que ResEdit™ ou Resorcerer™, copiez les ressources 'snd ' nécessaires dans la "resource fork" du fichier de structure de 4D.
• Transportez la base de Macintosh à Windows à l'aide de 4D Transporter.

Note importante : La version Windows de 4e Dimension ne joue pas les sons Macintosh compressés à l'aide de MACE. Si votre ressource 'snd' Macintosh ne se joue pas sous Windows, déterminez si le son est conforme aux conditions suivantes :

Champ de ressource snd	Valeur (en hexadécimal)
Version	0x0001
NbSynth	0x0001
SynthResID	0x0005
SynthInitOptions	0x000000A0
NbSoundCommand	0x0001
FirstCommand	0x8051

Vous pouvez vérifier les valeurs internes d'une ressource 'snd ' à l'aide de Resorcerer™.

Exemples

(1) L'exemple suivant montre comment jouer un fichier vidéo de votre choix sous Windows :

```
$DocRéf := Ouvrir document ( ""; "AVI")
Si (OK=1)
   FERMER DOCUMENT($DocRéf)
⇒  JOUER SON (Document)
Fin de si
```

(2) L'exemple suivant se trouve dans une méthode de démarrage. Ce son est joué lors de l'ouverture de la base sous MacOS :

```
⇒  JOUER SON ("Bienvenue")  ` Jouer le son de bienvenue
```

Référence

BEEP.

Lire interface

Interface utilisateur

version 6.7 (Modifiée)

Lire interface → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Interface de plate-forme en cours d'utilisation

Description

La fonction Lire interface retourne une valeur numérique indiquant l'interface de plate-forme utilisée pour afficher les formulaires.

Cette fonction peut retourner une des constantes prédéfinies suivantes :

Constante	Type	Valeur
Plate-forme automatique	Entier long	-1
Mac Système 7	Entier long	0
Windows NT 3.51	Entier long	1
Windows 9x	Entier long	2
Platinum	Entier long	3
Thème Mac	Entier long	4

Vous pouvez modifier l'interface de plate-forme à l'aide de la commande FIXER INTERFACE ou en utilisant la boîte de dialogue des **Propriétés de la base** en mode Structure.

Référence

FIXER INTERFACE.

FIXER INTERFACE (interface)

Paramètre	Type		Description
interface	Numérique	→	Nouvelle interface de plate-forme : -1 Plate-forme automatique 0 Mac Système 7 1 Windows NT 3.51 2 Windows 9x 3 Platinum 4 Thème Mac

Description

La commande FIXER INTERFACE définit l'interface de plate-forme utilisée pour afficher les formulaires.

Vous passez dans le paramètre interface une des constantes prédéfinies suivantes :

Constante	Type	Valeur
Plate-forme automatique	Entier long	-1
Mac Système 7	Entier long	0
Windows NT 3.51	Entier long	1
Windows 9x	Entier long	2
Platinum	Entier long	3
Thème Mac	Entier long	4

Note : La constante Thème Mac permet d'utiliser l'interface définie dans le *Gestionnaire d'apparence*. Ce gestionnaire n'existe que sous MacOS. Lorsqu'une base définie en Thème Mac est affichée sous Windows, l'interface "Windows 9x" est appliquée.

Si la valeur que vous passez ne modifie pas l'interface de plate-forme courante, la commande ne fait rien.

Note : L'interface de plate-forme peut également être modifiée dans la boîte de dialogue des **Propriétés de la base** en mode Structure.

Exemple

Avec 4D Server, vous voulez que les postes 4D Client Macintosh et Windows utilisent simultanément des interfaces de plate-formes différentes. Pour cela, vous pouvez appeler la commande FIXER INTERFACE dans la Méthode base Sur ouverture :

```
      ` Partons de l'hypothèse que les préférences des utilisateurs
      ` sont stockées dans une table [Préférences]
      ` Cherchons l'enregistrement correspondant à l'utilisateur courant
   CHERCHER([Préférences];[Préférences]Utilisateur=Utilisateur courant)
   Si (Enregistrements trouves([Préférences])=0)
         ` S'il n'est pas trouvé, utilisons les préférences par défaut
      CHERCHER([Préférences];[Préférences]Utilisateur="Par défaut")
   Fin de si
      ` Fixons l'interface de plate-forme en fonction des préférences de l'utilisateur
⇒  FIXER INTERFACE ([Préférences]Interface plate-forme)
```

Référence

Lire interface.

FIXER TITRES TABLES (titresTables; numTables)

Paramètre	Type		Description
titresTables	Tableau Alpha	→	Noms des tables tels qu'ils doivent apparaître
numTables	Tableau Num	→	Numéros des tables

Description

FIXER TITRES TABLES vous permet de masquer, renommer et réordonner les tables de votre base lorsqu'elles apparaissent dans les éditeurs standard de 4e Dimension, comme l'éditeur de recherches, en mode Utilisation ou Menus créés.
Cette commande vous permet également de modifier les libellés des tables apparaissant dans vos formulaires, si vous avez utilisé des libellés dynamiques. Pour plus d'informations sur l'insertion de libellés de tables et de champs dynamiques dans les formulaires, reportez-vous au manuel *Mode Structure*.

Les tableaux titresTables et numTables doivent être synchronisés. Dans le tableau titresTables, vous passez les noms des tables tels que vous voulez qu'ils apparaissent. Les tables s'afficheront dans l'ordre défini par ces tableaux. Dans chaque élément du tableau numTables, passez le numéro de la table qui correspond au nom, nouveau ou ancien, de la table, et ce dans le même numéro d'élément que dans le tableau titresTables. Si vous voulez masquer une table, ne la mettez pas dans les tableaux.

Vous avez, par exemple, une base comprenant les tables A, B et C, créées dans cet ordre. Vous voulez que ces tables apparaissent sous les noms X, Y et Z. De plus, vous ne voulez pas faire apparaître la table B. Enfin, vous voulez que les tables soient dans l'ordre Z et X. Pour cela, vous passez dans le paramètre titresTables un tableau à deux éléments, Z et X, et vous passez dans le paramètre numTables un tableau à deux éléments, 3 et 1.

FIXER TITRES TABLES ne modifie pas la structure de votre base. Cette commande n'affecte que l'utilisation ultérieure des éditeurs standard de 4e Dimension et des formulaires comportant des libellés dynamiques. L'aire de validité de la commande FIXER TITRES TABLES est la session. L'avantage, en Client/Serveur, est que plusieurs postes 4D Client peuvent “voir” simultanément votre structure d'une manière différente. Vous pouvez appeler FIXER TITRES TABLES autant de fois que vous voulez.

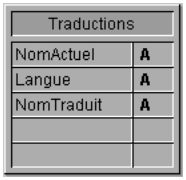
La commande FIXER TITRES TABLES est utile dans les cas suivants :
• Traduction dynamique de votre base.
• Affichage des tables dans l'ordre et avec les noms que vous voulez, indépendamment de leur définition.
• Affichage des tables d'une façon qui dépend de l'identité ou des privilèges d'un utilisateur.

ATTENTION : FIXER TITRES TABLES n'annule pas l'effet de la propriété Invisible d'une table. Si vous avez défini une table en tant qu'invisible au niveau de la structure, elle n'apparaîtra pas, même si elle est spécifiée dans FIXER TITRES TABLES.

Exemple

• Vous développez une application 4D destinée au marché international. Vous avez donc besoin de prendre en compte les nécessités de traduction et de localisation. Pour les éditeurs standard de 4e Dimension qui apparaissent en mode Utilisation ou Menus créés et vos formulaires utilisant des libellés dynamiques, vous pouvez traiter cette question en utilisant une table [Traductions] et quelques méthodes pour créer et utiliser les traductions pour chaque langue que vous voulez.

• Dans votre base, vous créez la table suivante :



Traductions	
NomActuel	A
Langue	A
NomTraduit	A

• Ensuite, créez la méthode projet traduire_TABLES_ET_CHAMPS ci-dessous. Cette méthode analyse la structure de votre base dans la table [Traductions] et crée les enregistrements correspondant à la langue passée comme paramètre.

```
` méthode projet traduire_TABLES_ET_CHAMPS
` traduire_TABLES_ET_CHAMPS ( Alpha )
` traduire_TABLES_ET_CHAMPS ( Langue )

C_ALPHA(31;$1)
C_ENTIER LONG($vlTable;$vlChamp)

Boucle ($vlTable;1;Nombre de tables) ` Passer sur chaque table
      ` Vérifier s'il existe une traduction du nom de la table pour la langue spécifiée
   CHERCHER([Traductions];[Traductions]NomActuel=Nom de la table($vlTable);*)
   CHERCHER([Traductions]; & ;[Traductions]Langue=$1)
```

```
Si (Enregistrements trouves([Traductions])=0)
      ` Sinon, créer l'enregistrement
   CREER ENREGISTREMENT([Traductions])
   [Traductions]NomActuel:=Nom de la table($vlTable)
   [Traductions]Langue:=$1
      ` Le nom de la table traduit a besoin d'être saisi
   STOCKER ENREGISTREMENT([Traductions])
Fin de si
Boucle ($vlChamp;1;Nombre de champs($vlTable))
      ` Vérifier s'il existe une traduction du nom du champ dans la langue spécifiée
   CHERCHER([Traductions];[Traductions]NomActuel=Nom du champ($vlTable;
                                                         $vlChamp);*)
   CHERCHER([Traductions]; & ;[Traductions]Langue=$1)
   Si (Enregistrements trouves([Traductions])=0)
         ` Sinon, créer l'enregistrement
      CREER ENREGISTREMENT([Traductions])
      [Traductions]NomActuel:=Nom du champ($vlTable;$vlChamp)
      [Traductions]Langue:=$1
         ` Le nom du champ traduit a besoin d'être saisi
      STOCKER ENREGISTREMENT([Traductions])
   Fin de si
Fin de boucle
Fin de boucle
```

• Si maintenant vous exécutez la ligne suivante, vous pouvez créer autant d'enregistrements qu'il vous faut pour la traduction espagnole de vos tables et champs :

```
traduire_TABLES_ET_CHAMPS ("Espagnol")
```

• Une fois que cette ligne de code est appelée, vous pouvez saisir une traduction dans le champ [Traductions]NomTraduit pour chacun des nouveaux enregistrements.

• Enfin, chaque fois que vous voulez afficher en espagnol les éditeurs standard de 4D ou les formulaires avec libellés dynamiques, vous exécutez la ligne suivante :

```
TABLES_ET_CHAMPS_LOCALISES ("Espagnol")
```

La méthode projet TABLES_ET_CHAMPS_LOCALISES est la suivante :

```
      ` Méthode objet TABLES_ET_CHAMPS_LOCALISES
      ` TABLES_ET_CHAMPS_LOCALISES ( Alpha )
      ` TABLES_ET_CHAMPS_LOCALISES ( Langue )

C_ALPHA(63;$1)
C_ENTIER LONG($vlTable;$vlNumTable;$vlChamp;$vlNumChamp)
```

```
$vlNumTable:=Nombre de tables  ` Obtenir le nombre de tables dans la base
   ` Initialiser les tableaux pour FIXER TITRES TABLES
TABLEAU ALPHA(31;$asNomTable;$vlNumTable)
TABLEAU ENTIER($aiNuméroTable;$vlNumTable)
Boucle ($vlTable;1;$vlNumTable)  ` Passer sur chaque table
   $asNomTable{$vlTable}:=Nom de la table($vlTable)  ` Obtenir le nom de la table
   $aiNuméroTable{$vlTable}:=$vlTable  ` Stocker le numéro de la table
      ` Chercher la traduction
   CHERCHER([Traductions];[Traductions]NomActuel=$asNomTable{$vlTable};*)
   CHERCHER([Traductions]; & ;[Traductions]Langue=$1)
   Si (Enregistrements dans table([Traductions])>0)
         ` Si disponible, utiliser le nom localisé de la table
      $asNomTable{$vlTable}:=[Traductions]NomTraduit
   Fin de si
      ` Obtenir le nombre de champs dans la table
   $vlNumChamp:=Nombre de champs($vlTable)
      ` Initialiser les tableaux pour FIXER TITRES CHAMPS
   TABLEAU ALPHA(31;$asNomChamp;$vlNumTable)
   TABLEAU ENTIER($asNumChamp;$vlNumTable)
   Boucle ($vlChamp;1;$vlNumChamp)  ` Pour chaque champ
         ` Obtenir le nom du champ
      $asNomChamp{$vlChamp}:=Nom du champ($vlTable;$vlChamp)
      $asNumChamp{$vlChamp}:=$vlChamp  ` Stocker le numéro du champ
      CHERCHER([Traductions];[Traductions]NomActuel=$asNomChamp{$vlChamp};*)
         ` Chercher la traduction
      CHERCHER([Traductions]; & ;[Traductions]Langue=$1)
      Si (Enregistrements dans table([Traductions])>0)
            ` Si disponible, utiliser le nom traduit
         $asNomChamp{$vlChamp}:=[Traductions]NomTraduit
      Fin de si
   Fin de boucle
   TRIER TABLEAU($asNomChamp;$asNumChamp;>)
⇒  FIXER TITRES CHAMPS(Table($vlTable)->;$asNomChamp;$asNumChamp)
Fin de boucle
TRIER TABLEAU($asNomTable;$aiNuméroTable;>)
⇒  FIXER TITRES TABLES($asNomTable;$aiNuméroTable)
```

• Notez que de nouvelles traductions peuvent être effectuées dans la base sans modification de code ni recompilation.

Référence

FIXER TITRES CHAMPS, Nom de la table, Nombre de tables.

FIXER TITRES CHAMPS (table | sous-table; titresChamps; numChamps)

Paramètre	Type		Description
table | sous-table	Table | Sous-table	→	Table ou sous-table pour laquelle vous voulez redéfinir les titres des champs
titresChamps	Tableau Alpha	→	Nouveaux titres des champs
numChamps	Tableau Num	→	Numéros des champs

Description

FIXER TITRES CHAMPS vous permet de masquer, renommer et réordonner les champs d'une table ou d'une sous-table de votre base lorsqu'ils apparaissent dans les éditeurs standard de 4e Dimension, tel que l'éditeur de recherches, en mode Utilisation ou Menus créés.
Cette commande vous permet également de modifier les libellés des champs apparaissant dans vos formulaires, si vous avez utilisé des libellés dynamiques. Pour plus d'informations sur l'insertion de libellés de tables et de champs dynamiques dans les formulaires, reportez-vous au manuel *Mode Structure*.

Les tableaux titresChamps et numChamps doivent être synchronisés. Dans le tableau titresChamps, vous passez les noms des champs tels que vous voulez qu'ils apparaissent. Les champs s'afficheront dans l'ordre défini par ces tableaux. Dans chaque élément du tableau numChamps, vous passez le numéro de la table qui correspond au nom, nouveau ou ancien, du champ, et ce dans le même numéro d'élément que dans le tableau titresChamps. Si vous voulez masquer un champ, il suffit de ne pas le passer dans les tableaux.

Vous avez, par exemple, une table comprenant les champs F, G et H, créés dans cet ordre. Vous voulez que ces champs apparaissent comme M, N et O. De plus, vous ne voulez pas faire apparaître le champ N. Enfin, vous voulez que les tables soient dans l'ordre O et M. Pour cela, vous passez dans le paramètre titresChamps un tableau contenant deux éléments, O et M, et vous passez dans le paramètre numChamps un tableau contenant deux éléments, 3 et 1.

FIXER TITRES CHAMPS ne modifie pas la structure de votre base. Elle n'affecte que l'affichage ultérieur des éditeurs standard de 4e Dimension et des formulaires comportant des libellés dynamiques. Le temps de validité de la commande FIXER TITRES CHAMPS est la session. L'avantage, en Client/Serveur, est que plusieurs stations 4D Client peuvent “voir” simultanément votre structure d'une manière différente. Vous pouvez appeler FIXER TITRES CHAMPS autant de fois que vous voulez.

La commande FIXER TITRES CHAMPS est utile dans les cas suivants :
• Traduction dynamique de votre base.
• Affichage des champs dans l'ordre et avec les noms que vous voulez, indépendamment de leurs définitions.
• Affichage des champs suivant l'identité ou les privilèges d'un utilisateur.

ATTENTION :
• FIXER TITRES CHAMPS n'annule pas l'effet de la propriété Invisible d'un champ. Si vous avez défini un champ en tant qu'invisible au niveau de la structure, il n'apparaîtra pas même s'il est spécifié dans FIXER TITRES CHAMPS.
• FIXER TITRES CHAMPS doit impérativement être accompagnée d'un appel à la commande FIXER TITRES TABLES, même si vous ne souhaitez pas modifier le titre de la table. Si vous exécutez la commande FIXER TITRES CHAMPS seule, 4D ne fait rien.

Exemple

Reportez-vous à l'exemple de la commande FIXER TITRES TABLES.

Référence

FIXER TITRES TABLES, Nom du champ, Nombre de champs.

Majuscule enfoncee → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Etat de la touche Majuscule

Description

Majuscule enfoncee retourne Vrai si la touche **Majuscule** est enfoncée.

Exemple

La méthode objet du bouton bUnBouton effectue des actions différentes en fonction de la ou des touche(s) de modification enfoncée(s) au moment du clic :

```
   ` Méthode objet bUnBouton
Au cas ou
      ` Diverses autres combinaisons de touches peuvent être testées ici
      ` ...
   : (Majuscule enfoncee & Windows Ctrl enfoncee)
      ` Les touches Maj et Ctrl Windows (ou Commande MacOS) sont enfoncées
      FAIRE ACTION1
      ` ...
   : (Majuscule enfoncee)
      ` Seule Majuscule est enfoncée
      FAIRE ACTION2
      ` ...
   : (Windows Ctrl enfoncee)
      ` Seule Ctrl Windows (ou Commande MacOS) est enfoncée
      FAIRE ACTION3
      ` ...
      ` D'autres touches peuvent être testées individuellement ici
      ` ...
Fin de cas
```

Référence

Macintosh commande enfoncee, Macintosh control enfoncee, Macintosh option enfoncee, Verrouillage majuscule enfoncee, Windows Alt enfoncee, Windows Ctrl enfoncee.

Verrouillage majuscule enfoncee → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Etat de la touche Verrouillage Majuscule

Description

Verrouillage majuscule enfoncee retourne Vrai si la touche **Verrouillage Majuscule** est enfoncée.

Exemple

Reportez-vous à l'exemple de la commande Majuscule enfoncee.

Référence

Macintosh commande enfoncee, Macintosh control enfoncee, Macintosh option enfoncee, Majuscule enfoncee, Windows Alt enfoncee, Windows Ctrl enfoncee.

Windows Ctrl enfoncee → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Etat de la touche Ctrl Windows ou Etat de la touche Commande Macintosh

Description

Windows Ctrl enfoncee retourne Vrai si la touche **Ctrl** Windows est enfoncée.

Note : Lorsqu'elle est appelée sous MacOS, Windows Ctrl enfoncee retourne Vrai si la touche Macintosh **Commande** est enfoncée.

Exemple

Reportez-vous à l'exemple de la commande Majuscule enfoncee.

Référence

Macintosh commande enfoncee, Macintosh option enfoncee, Majuscule enfoncee, Verrouillage majuscule enfoncee, Windows Alt enfoncee, Windows Ctrl enfoncee.

Windows Alt enfoncee → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Etat de la touche Windows Alt ou Etat de la touche Macintosh Option

Description

Windows Alt enfoncee retourne Vrai si la touche **Alt** Windows est enfoncée.

Note : Lorsqu'elle est appelée sous MacOS, Windows Alt enfoncee retourne Vrai si la touche Macintosh **Option** est enfoncée.

Exemple

Reportez-vous à l'exemple de la commande Majuscule enfoncee.

Référence

Macintosh commande enfoncee, Macintosh control enfoncee, Macintosh option enfoncee, Majuscule enfoncee, Verrouillage majuscule enfoncee, Windows Ctrl enfoncee.

Macintosh commande enfoncee → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Etat de la touche Commande Macintosh ou Etat de la touche Ctrl Windows

Description

Macintosh commande enfoncee retourne Vrai si la touche **Commande** Macintosh est enfoncée.

Note : Lorsqu'elle est appelée sous Windows, la fonction Macintosh commande enfoncee retourne Vrai si la touche **Ctrl** Windows est enfoncée.

Exemple

Reportez-vous à l'exemple de la commande Majuscule enfoncee.

Référence

Macintosh control enfoncee, Macintosh option enfoncee, Majuscule enfoncee, Verrouillage majuscule enfoncee, Windows Alt enfoncee, Windows Ctrl enfoncee.

Macintosh option enfoncee → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Etat de la touche Option Macintosh ou Etat de la touche Alt Windows

Description

Macintosh option enfoncee retourne Vrai si la touche **Option** Macintosh est enfoncée.

Note : Lorsqu'elle est appelée sous Windows, la fonction Macintosh option enfoncee retourne Vrai si la touche **Alt** Windows est enfoncée.

Exemple

Reportez-vous à l'exemple de la commande Majuscule enfoncee.

Référence

Macintosh commande enfoncee, Macintosh control enfoncee, Majuscule enfoncee, Verrouillage majuscule enfoncee, Windows Alt enfoncee, Windows Ctrl enfoncee.

Macintosh control enfoncee → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Etat de la touche Control du Macintosh

Description

Macintosh control enfoncee retourne Vrai si la touche **Control** du Macintosh est enfoncée.

Note : Lorsqu'elle est appelée sous Windows, la fonction Macintosh control enfoncee retourne toujours Faux. Cette touche Macintosh n'a pas d'équivalent sous Windows.

Exemple

Reportez-vous à l'exemple de la commande Majuscule enfoncee.

Référence

Macintosh commande enfoncee, Macintosh option enfoncee, Majuscule enfoncee, Verrouillage majuscule enfoncee, Windows Alt enfoncee, Windows Ctrl enfoncee.

POSITION SOURIS (sourisX; sourisY; boutonSouris{; *})

Paramètre	Type		Description
sourisX	Numérique	←	Coordonnée horizontale de la souris
sourisY	Numérique	←	Coordonnée verticale de la souris
boutonSouris	Numérique	←	Etat du bouton de la souris : 0 = Bouton relâché 1 = Bouton enfoncé 2 = Bouton droit enfoncé (Windows seulement) 3 = Les deux boutons enfoncés (Windows seulement)
*		→	Si spécifié, utiliser le système de coordonnées globales Si omis, utiliser le système de coordonnées locales

Description

La commande POSITION SOURIS retourne l'état courant de la souris.

Les coordonnées horizontale et verticale sont retournées dans les paramètres sourisX et sourisY. Si vous passez le paramètre *, ces coordonnées sont exprimées par rapport à l'écran. Si vous ne passez pas le paramètre *, vous exprimez ces coordonnées par rapport à la fenêtre de premier plan du process courant.

Le paramètre boutonSouris retourne l'état du ou des bouton(s) de la souris, comme décrit ci-dessus, dans le tableau des paramètres.

Exemple

Reportez-vous à l'exemple de la commande Pop up menu.

Référence

APPELER SUR EVENEMENT, Macintosh commande enfoncee, Macintosh control enfoncee, Macintosh option enfoncee, Majuscule enfoncee, Verrouillage majuscule enfoncee, Windows Alt enfoncee, Windows Ctrl enfoncee.

Pop up menu (contenu{; parDéfaut}) → Numérique

Paramètre	Type		Description
contenu	Texte	→	Définition du texte du menu
parDéfaut	Numérique	→	Numéro de l'élément sélectionné par défaut
Résultat	Numérique	←	Numéro de l'élément de menu sélectionné

Description

La commande Pop up menu fait apparaître un pop up à l'emplacement courant du curseur de la souris.

Selon les règles standard d'interface utilisateur, cette commande doit généralement être appelée en réponse à un clic souris, et lorsque le bouton reste enfoncé un certain laps de temps.

Vous définissez les éléments du pop up menu à l'aide du paramètre contenu, de la manière suivante :
• Chaque élément est séparé des autres par un point-virgule (;), "Elément1;Elément2;Elément3".
• Pour inactiver un élément, placez une parenthèse ouvrante "(" dans son libellé.
• Pour définir une ligne de séparation, passez la valeur "(-" en tant que libellé.
• Pour définir le style de caractères d'un élément, placez dans son libellé le symbole inférieur à "<" suivi d'une lettre. Voici les différents codes :

<B	Gras
<I	Italique
<U	Souligné
<O	Contours (MacOS seulement)
<S	Relief (MacOS seulement)

• Pour associer une coche à un élément, insérez dans son libellé un point d'exclamation "!" suivi du caractère que vous voulez utiliser comme coche. Sous MacOS, le caractère est affiché ; sous Windows, une coche standard est affichée (quel que soit le caractère passé).
• Pour associer une icône à un élément, insérez dans son libellé un accent circonflexe "^" suivi d'un caractère dont le code ASCII moins 48 plus 256 (ou plus 208) représente un numéro de ressource d'icône MacOS.

• Pour ajouter un raccourci clavier à un élément, insérez dans son libellé une barre oblique "/" suivie du caractère de raccourci. Notez que cette dernière option est uniquement informative (aucun raccourci clavier n'active le pop up menu), cependant vous pouvez indiquer un raccourci clavier si l'élément du pop up menu dispose d'une commande équivalente dans la barre de menus principale de votre application.

Le paramètre optionnel parDéfaut vous permet de définir l'élément du pop up menu sélectionné par défaut lorsque celui-ci apparaît. Passez une valeur située entre 1 et le nombre d'éléments du menu. Si vous ne passez pas ce paramètre, le premier élément du menu sera sélectionné par défaut.

Lorsqu'un élément du pop up menu est sélectionné, la commande retourne son numéro, autrement elle retourne zéro.

Note : Utilisez les pop up menus avec un nombre "raisonnable" d'éléments. Si, par exemple, vous voulez afficher plus de 50 éléments, envisagez plutôt d'employer une zone de défilement dans un formulaire.

Exemple

La méthode projet MON RACCOURCI fait apparaître un pop up menu de navigation :

```
    ` Méthode projet MON RACCOURCI
POSITION SOURIS($vlMouseX;$vlMouseY;$vlBouton)
Si (Macintosh control enfoncee | ($vlBouton=2))
   $vtItems:="A propos de cette base...<l;(-;!-Autres options;(-"
   Boucle ($vlTable;1;Nombre de tables)
      $vtItems:=$vtItems+";"+Nom de la table($vlTable)
   Fin de boucle
⇒  $vlChoixUtilisateur:=Pop up menu($vtItems)
   Au cas ou
      : ($vlChoixUtilisateur=1)
         ` Afficher les informations
      : ($vlChoixUtilisateur=2)
         ` Afficher les options
      Sinon
         Si ($vlChoixUtilisateur>0)
            ` Aller à la table dont le numéro est $vlChoixUtilisateur-4
         Fin de si
   Fin de cas
Fin de si
```

Cette méthode projet peut être appelée d'une des manières suivantes :
• Depuis la méthode d'un objet réagissant à un clic souris, et n'attendant pas que le bouton soit relâché (par exemple un bouton invisible),
• Depuis un process qui “épie” les événements et communique avec les autres process,
• Depuis une méthode de gestion d'événements installée par la commande APPELER SUR EVENEMENT.

Dans les deux derniers cas, il n'est pas nécessaire que le clic se produise dans un objet de formulaire. C'est l'un des avantages de la commande Pop up menu. Généralement, les pop up menus sont affichés par l'intermédiaire d'objets de formulaire. Avec Pop up menu, vous pouvez faire apparaître un pop up menu n'importe où.

Le pop up menu s'affiche sous Windows lorsque l'utilisateur appuie sur le **bouton droit** de la souris, et sous MacOS lorsqu'il utilise la combinaison **Control+clic**. Notez cependant que la méthode ci-dessus ne teste pas le clic souris, c'est la méthode appelante qui en est chargée.

Voici le pop up menu tel qu'il s'affiche sous Windows (à gauche) et sous MacOS (à droite). Notez la coche standard de la version Windows :

Référence
POSITION SOURIS.

GENERER FRAPPE CLAVIER (code{; modifiers{; process}})

Paramètre	Type		Description
code	Numérique	→	Code ASCII d'un caractère ou code de touche de fonction
modifiers	Numérique	→	Etat des touches Modifier
process	Numérique	→	Numéro de référence du process de destination ou File d'attente des événements de l'application si ce paramètre est omis ou est égal à 0

Description

La commande GENERER FRAPPE CLAVIER simule la frappe d'une touche sur le clavier. Elle produit les mêmes effets que lorsque l'utilisateur tape réellement un caractère au clavier.

Vous passez le code ASCII du caractère dans le paramètre code.

Si vous n'utilisez pas le paramètre modifiers, aucun "modifier" (Majuscule, Option, etc...) n'est simulé. Si vous utilisez le paramètre modifiers, vous devez passer une constante ou une combinaison de constantes du thème **Evénements (Modifiers)**. Par exemple, pour simuler la touche Majuscule, passez la valeur Masque touche majuscule.

Si vous passez le paramètre process, la frappe clavier est envoyée au process dont le numéro de référence est spécifié. Si vous passez 0 (zéro) dans ce paramètre ou si vous l'omettez, la frappe clavier est envoyée au niveau de l'application et le gestionnaire de 4D l'affectera au process approprié.

Exemple

Reportez-vous à l'exemple de la fonction Chercher process.

Référence

GENERER CLIC, GENERER EVENEMENT.

GENERER CLIC (sourisX; sourisY{; process}{; *})

Paramètre	Type		Description
sourisX	Numérique	→	Coordonnée horizontale
sourisY	Numérique	→	Coordonnée verticale
process	Numérique	→	Numéro de référence du process de destination ou File d'attente des événements de l'application si ce paramètre est omis ou si vous passez 0
*		→	Si spécifié, utiliser le système de coordonnées globales Si omis, utiliser le système de coordonnées locales

Description

La commande GENERER CLIC simule un clic souris. Elle produit les mêmes effets que lorsque l'utilisateur clique réellement avec le bouton de la souris.

Vous passez les coordonnées horizontale et verticale du clic dans sourisX et sourisY. Si vous passez le paramètre *, vous exprimez ces coordonnées par rapport à l'écran. Si vous omettez le paramètre *, vous exprimez ces coordonnées par rapport à la fenêtre de premier plan du process dont le numéro est passé dans process.

Si vous passez le paramètre process, le clic est envoyé au process dont vous avez passé le numéro. Si vous passez 0 (zéro) ou si vous omettez ce paramètre, le clic est envoyé au niveau de l'application et le gestionnaire de 4D l'affectera au process approprié.

Référence

GENERER EVENEMENT, GENERER FRAPPE CLAVIER.

GENERER EVENEMENT (quoi; message; quand; sourisX; sourisY; modifiers{; process})

Paramètre	Type		Description
quoi	Numérique	→	Type d'événement
message	Numérique	→	Message de l'événement
quand	Numérique	→	Moment de l'événement exprimé en ticks
sourisX	Numérique	→	Coordonnée horizontale de la souris
sourisY	Numérique	→	Coordonnée verticale de la souris
modifiers	Numérique	→	Etat des touches Modifier
process	Numérique	→	Numéro de référence du process de destination ou File d'attente des événements de l'application si ce paramètre est omis ou si vous passez 0

Description

La commande GENERER EVENEMENT simule un événement (clavier ou souris). Elle produit les mêmes effets que lorsque l'utilisateur agit réellement par l'intermédiaire du clavier ou de la souris.

Vous devez passer une des constantes prédéfinies suivantes dans le paramètre quoi :

Constante	Type	Valeur
Bouton souris enfoncé	Entier long	1
Bouton souris relâché	Entier long	2
Touche enfoncée	Entier long	3
Touche relâchée	Entier long	4
Répétition touche	Entier long	5

Si l'événement est lié à la souris, passez 0 (zéro) dans le paramètre message. Si l'événement est lié au clavier, passez dans message le code ASCII du caractère simulé.

Généralement, vous passez la valeur retournée par la fonction Nombre de ticks dans quand.

Si l'événement est lié à la souris, passez les coordonnées horizontale et verticale du clic dans sourisX et sourisY.

Dans le paramètre modifiers, vous devez passer une constante ou une combinaison de constantes du thème **Evénements (Modifiers)**. Par exemple, pour simuler la touche Majuscule, passez la valeur Bit touche majuscule.

Si vous passez le paramètre process, l'événement est envoyé au process dont vous avez passé le numéro. Si vous passez 0 (zéro) ou si vous omettez ce paramètre, l'événement est envoyé au niveau de l'application et le gestionnaire de 4D l'affectera au process approprié.

Référence

GENERER CLIC, GENERER FRAPPE CLAVIER.

TEXTE SELECTIONNE (zone; débutSél; finSél)

Paramètre	Type		Description
zone	Champ | Variable	→	Champ ou variable saisissable
débutSél	Numérique	←	Position du début de la sélection de texte
finSél	Numérique	←	Position de la fin de la sélection de texte

Description

La commande TEXTE SELECTIONNE vous permet de déterminer précisément le texte actuellement sélectionné.

Attention : Bien que vous deviez passer à TEXTE SELECTIONNE un nom de variable ou de champ saisissable, cette commande ne retourne une position de sélection significative que lorsqu'elle est appliquée à la zone en cours de modification.

Note : Cette commande ne peut pas être utilisée avec des champs situés dans le formulaire en mode liste d'un sous-formulaire.

Le texte peut être sélectionné par l'utilisateur ou par SELECTIONNER TEXTE.

Le paramètre débutSél retourne la position du premier caractère sélectionné.
Le paramètre finSél retourne la position du dernier caractère sélectionné plus un.
Si les valeurs débutSél et finSél retournées sont identiques, l'utilisateur n'a pas sélectionné de texte et le point d'insertion est placé devant le caractère spécifié par débutSél.

Exemples

(1) L'exemple suivant récupère le texte sélectionné dans le champ [Produits]Notes :

```
⇒    TEXTE SELECTIONNE ([Produits]Notes;vPremier;vDernier)
     Si (vPremier<vDernier)
        ALERTE("Le texte sélectionné est : "+Sous chaine([Produits]Notes;vPremier;
                                                    vDernier-vPremier))
     Fin de si
```

(2) Reportez-vous à l'exemple de la commande FILTRER FRAPPE CLAVIER.

Référence

FILTRER FRAPPE CLAVIER, Frappe clavier, SELECTIONNER TEXTE.

SELECTIONNER TEXTE (zone; débutSél; finSél)

Paramètre	Type		Description
zone	Champ | Variable	→	Champ ou variable saisissable
débutSél	Numérique	→	Nouvelle position de début de sélection de texte
finSél	Numérique	→	Nouvelle position de fin de sélection de texte

Description

La commande SELECTIONNER TEXTE sélectionne une partie du texte dans zone.

Si zone n'est pas l'objet en cours de modification, elle récupère le focus.

Note : Cette commande ne peut pas être utilisée avec des champs situés dans le formulaire en mode liste d'un sous-formulaire.

Le paramètre débutSél représente la position du premier caractère à sélectionner, et le paramètre finSél représente la position du dernier caractère à sélectionner plus un. Si débutSél et finSél sont identiques, le point d'insertion est placé devant le caractère spécifié par débutSél et aucun caractère n'est sélectionné.

Si finSél est supérieur au nombre de caractères présents dans la zone, tous les caractères compris entre débutSél et la fin du texte sont sélectionnés.

Exemples

(1) L'exemple suivant sélectionne tous les caractères dans le champ saisissable [Produits]Notes :

```
⇒ SELECTIONNER TEXTE([Produits]Notes;1;Longueur([Produits]Notes)+1)
```

(2) L'exemple suivant place le point d'insertion au début du champ [Produits]Notes :

```
⇒ SELECTIONNER TEXTE([Produits]Notes;1;1)
```

(3) L'exemple suivant place le point d'insertion à la fin du champ [Produits]Notes :

```
   $vLen:=Longueur([Produits]Notes)+1
⇒ SELECTIONNER TEXTE([Produits]Notes;$vLen;$vLen)
```

(4) Reportez-vous à l'exemple de la commande FILTRER ENTREE CLAVIER.

CHANGER POINTEUR SOURIS {(curseur)}

Paramètre	Type		Description
curseur	Numérique	→	Numéro de ressource MacOS de curseur

Description

La commande CHANGER POINTEUR SOURIS remplace le pointeur (graphique) de la souris par celui qui est stocké dans la ressource type MacOS 'CURS' dont vous avez passé le numéro d'ID dans le paramètre curseur.

Si vous ne passez pas ce paramètre, le pointeur de la souris (re)devient la flèche standard.

Utilisez la commande LISTE RESSOURCES pour récupérer la liste des curseurs disponibles.

Référence

LISTE RESSOURCES.

Dernier objet → Pointeur

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Pointeur	←	Pointeur vers la dernière zone de saisie ou la zone de saisie courante

Description

Dernier objet retourne un pointeur vers la dernière zone de saisie (champ ou variable) modifiée ou vers la zone de saisie courante ; autrement dit, vers l'objet dans lequel se trouve ou se trouvait en dernier lieu le curseur. Vous pouvez utiliser Dernier objet pour effectuer une action dans un formulaire sans savoir quel objet est actuellement sélectionné. N'oubliez pas auparavant de tester si l'objet est du type voulu, à l'aide de la fonction Type. Cette commande ne peut pas être utilisée sur les champs dans les sous-formulaires.

Note : Cette commande n'a de sens qu'en cours de saisie. Son utilisation hors de ce contexte génère des messages d'erreur.

Exemple

L'exemple suivant est une méthode objet pour un bouton. Cette méthode passe les données de l'objet courant en majuscules. L'objet doit être de type Texte ou Alpha (type 0 ou 24):

```
⇒    $pointeur := Dernier objet  ` Obtenir le pointeur vers le dernier objet
     Si ((Type ($pointeur->) = Est un champ alpha) | (Type($pointeur->) =
                                                        Est une variable chaîne))
           ` S'il s'agit d'un objet de type Texte ou Alpha
        $pointeur>> := Majusc ($pointeur>>)  ` Mettre les données en majuscules
     Fin de si
```

REDESSINER (objet)

Paramètre	Type		Description
objet	Objet	→	Sous-table de laquelle redessiner le sous-formulaire ou Table de laquelle redessiner le sous-formulaire ou Champ duquel redessiner la zone ou Variable de laquelle redessiner la zone ou Formulaire 4D à mettre à jour sur les browsers Web

Description

Lorsque vous modifiez par programmation le contenu d'un champ ou d'un sous-champ affiché dans un sous-formulaire, vous devez exécuter la commande REDESSINER pour vous assurer que le formulaire est correctement mis à jour.

Note Serveur Web : Lorsqu'elle est associée à l'événement formulaire Sur minuteur, la commande REDESSINER permet de provoquer la mise à jour d'un formulaire 4D affiché par un browser Web. Pour plus d'informations sur ce point, reportez-vous à la description de la commande FIXER MINUTEUR.

Référence

FIXER MINUTEUR.

INVERSER FOND ({*; }textVar | textChp)

Paramètre	Type		Description
*	*	→	Si spécifié, on passe le nom de l'objet (chaîne) Si omis, on passe un champ ou une variable
textVar | textChp	Variable | Champ	→	Variable ou champ de type texte dont le fond doit être inversé

Description

INVERSER FOND permet d'inverser textVar ou textChp dans un formulaire. La portée de cette commande est le formulaire en cours d'utilisation.

Si vous passez le paramètre optionnel *, vous indiquez que le second paramètre désigne le nom d'un objet (une chaîne). Si vous ne passez pas *, le paramètre désigne un champ ou une variable. Dans ce cas, vous passez la référence du champ ou de la variable texte.

Vous pouvez utiliser INVERSER FOND lors d'un affichage à l'écran ou lors d'une impression sur une imprimante matricielle. Une imprimante postscript ne permet pas d'inverser le fond. Vous ne pouvez pas non plus inverser de variable dans un formulaire sortie. Evitez d'utiliser INVERSER FOND avec une variable saisissable : la saisie de caractères effacera partiellement seulement l'inversion.

Exemple

L'exemple suivant est la méthode objet d'une variable dans un formulaire entrée. La méthode teste la valeur d'un champ. Si elle est négative, la variable est inversée dans le formulaire :

```
   vMontant := [Comptes]Montant ` Assignation de la valeur du champ à la variable
   Si (vMontant < 0) ` Si le montant est négatif...
⇒     INVERSER FOND (vMontant) ` Inverser le fond de la variable
   Fin de si
```

Note : Cette commande, créée à l'origine pour les interfaces en noir et blanc, est désormais rarement utilisée. Pour signaler ou mettre en avant un champ ou une variable dont la valeur est par exemple incorrecte, un système de couleurs est généralement utilisé.

Référence

CHOIX COULEUR, FIXER COULEURS RVB.

29

Interruptions

APPELER SUR EVENEMENT

Interruptions

version 5

APPELER SUR EVENEMENT (méthodeEvén{; nomProcess})

Paramètre	Type		Description
méthodeEvén	Alpha	→	Méthode d'événement à appeler ou Chaîne vide pour arrêter l'interception des événements
nomProcess	Alpha	→	Nom de process

Description

APPELER SUR EVENEMENT installe la méthode dont le nom est passé dans méthodeEvén comme méthode de gestion des événements.

Conseil : Cette commande nécessite un niveau de connaissances avancé en programmation. Généralement, vous n'avez pas besoin d'appeler APPELER SUR EVENEMENT pour traiter les événements. Lorsque vous utilisez des formulaires, 4e Dimension gère pour vous les événements et les retourne aux formulaires et objets appropriés.

Astuce : La version 6 a introduit de nouvelles commandes telles que POSITION SOURIS, Majuscule enfoncee, etc., pour récupérer des informations sur les événements. Ces commandes, dans une certaine mesure, peuvent être appelées depuis les méthodes objet pour traiter les informations dont vous avez besoin. Elles peuvent ainsi vous épargner l'écriture d'un algorithme basé sur une structure du type APPELER SUR EVENEMENT.

La portée de cette commande est la session de travail. Par défaut, la méthode est exécutée dans un process local séparé. Vous ne pouvez avoir qu'une méthode de gestion d'événement à la fois. Pour désinstaller une méthode de gestion d'événement, appelez de nouveau APPELER SUR EVENEMENT et passez une chaîne vide dans méthodeEvén.

Comme la méthode de gestion d'événement tourne dans process séparé, méthodeEvén est toujours active, même si aucune méthode 4e Dimension n'est en cours d'exécution. Après l'installation, 4e Dimension appelle la méthode méthodeEvén dès qu'un événement survient. Un événement peut être un clic souris ou la frappe d'une touche.

Le paramètre optionnel nomProcess permet de donner un nom au process créé par APPELER SUR EVENEMENT. Si nomProcess commence par le symbole dollar ($), nomProcess est un process local, ce dont vous aurez généralement besoin. Si vous ne passez pas le paramètre nomProcess, 4D crée par défaut un process local nommé $Gestionnaire d'événement.

ATTENTION : Soyez prudent lors de l'écriture d'une méthode de gestion d'événement. N'appelez pas de commande générant un événement, sinon vous risquez de ne plus pouvoir sortir de la méthode. La combinaison de touches **Ctrl**+**Maj**+**Retour Arrière** (sous Windows) ou **Commande**+**Option**+**Maj**+**Control**+**Retour Arrière** (sous MacOS) convertit le process d'événement en un process normal, ce qui signifie que la méthode ne reçoit plus systématiquement tous les événements qui surviennent. Cette combinaison vous permet de sortir d'une méthode de gestion d'événement devenue incontrôlable.

Dans la méthode de gestion d'événement, vous pouvez lire les variables système suivantes : MouseDown, KeyCode, Modifiers, MouseX, MouseY et MouseProc. Notez que ces variables sont des variables process. Leur portée est donc le process de gestion d'événements. Copiez-les dans des variables interprocess si vous souhaitez que leurs valeurs soient disponibles dans un autre process.

• La variable système MouseDown contient 1 s'il y a eu un clic souris, 0 sinon.

• La variable système KeyCode contient le code ASCII du caractère tapé au clavier, ou le code d'une touche de fonction. Référez-vous à la section Codes ASCII qui liste les codes ASCII utilisés par 4D, ainsi qu'à la section Codes des touches de fonction. 4D fournit des constantes prédéfinies pour les principaux codes ASCII et touches de fonctions. Vous pouvez les visualiser à l'aide la fenêtre de l'Explorateur, dans les thèmes correspondants.

• La variable système Modifiers permet de savoir si les touches suivantes étaient enfoncées lorsqu'un événement s'est produit :

Plate-forme	**Modifiers**
Windows	Maj, Verrouillage des majuscules, Alt, Ctrl, Bouton droit de la souris
Macintosh	Maj, Verrouillage des majuscules, Option, Contrôle, Commande

Notes
- la touche Windows Alt est l'équivalent de la touche Macintosh Option.
- la touche Windows Ctrl est l'équivalent de la touche Macintosh Commande.
- La touche Macintosh Control n'a pas d'équivalent sous Windows. Cependant, un clic bouton droit de la souris sous Windows est l'équivalent de Control+clic sur Macintosh.

Isolément, les touches "modifiers" ne génèrent pas d'événement. Une autre touche ou le bouton de la souris doit également être enfoncé(e). La variable Modifiers est une variable de type Entier long (4 octets), qui doit être considérée comme un tableau de 32 bits. 4D fournit des constantes prédéfinies exprimant la position ou le masque des bits pour tester le bit correspondant à chaque touche de modification. Lorsque, par exemple, vous voulez détecter si la touche Majuscule a été enfoncée pour l'événement, vous pouvez écrire :

```
Si (Modifiers ?? Bit touche majuscule)  ` Si la touche Majuscule était enfoncée
```

ou :

```
Si ((Modifiers & Masque touche majuscule)#0) ` Si la touche Majuscule était enfoncée
```

• Les variables systèmes MouseX et MouseY contiennent les coordonnées horizontale et verticale du clic souris, exprimées dans le système de coordonnées locales de la fenêtre dans laquelle le clic s'est produit. L'angle supérieur gauche de la fenêtre représente les coordonnées 0,0. Ces variables n'ont de signification que lorsqu'un clic souris a lieu.

• La variable système MouseProc contient le numéro de référence du process dans lequel le clic souris s'est produit.

Note : Les variables système MouseDown, KeyCode, Modifiers, MouseX, MouseY et MouseProc ne contiennent des valeurs significatives que dans une méthode de gestion d'événement installée par APPELER SUR EVENEMENT.

Exemple

L'exemple suivant annule l'impression si l'utilisateur appuie sur les touches **Ctrl**+**.** (**Commande**+**.**). En premier lieu, la méthode de gestion des événements est installée. Ensuite, un message s'affiche, indiquant que l'impression a été annulée. Si la variable interprocess <>vbOnStoppe est égale à Vrai dans la méthode de gestion d'événement, une boîte de dialogue d'alerte s'affiche pour indiquer à l'utilisateur le nombre d'enregistrements qui viennent de s'imprimer. Enfin, la méthode de gestion d'événement est désinstallée :

```
UTILISER PARAMETRES IMPRESSION
Si (OK =1)
   <>vbOnStoppe:=Faux
⇒  APPELER SUR EVENEMENT("GESTION EVENEMENTS")
   TOUT SELECTIONNER([Personnes])
   MESSAGE("Pour interrompre l'impression, appuyez sur Ctrl+point.")
   $NbEnregistrements:=Enregistrements trouves([Personnes])
```

```
      Boucle (Enrg;1;NbEnregistrements)
         Si (<>vbOnStoppe)
            ALERTE("L'impression a été annulée à l'enregistrement "+Chaine(Enrg)+
                                                    " sur "+Chaine(NbEnregistrements))
            Enrg:=NbEnregistrements+1
         Sinon
            IMPRIMER LIGNE([Personnes];"Etat")
         Fin de si
      Fin de boucle
      SAUT DE PAGE
         ` Désinstallation de la méthode d'appel sur événement
⇒     APPELER SUR EVENEMENT("")
   Fin de si
```

La méthode de gestion d'événement teste si la combinaison de touches **Ctrl**+**.** (**Commande**+**.**) a été employée et met la variable interprocess <>vbOnStoppe à Vrai :

```
` Méthode projet GESTION EVENEMENTS
Si ((Modifiers ?? Bit touche commande) & ( KeyCode = Point))
   CONFIRMER("Voulez-vous vraiment annuler l'impression ?")
   Si (OK=1)
      <>vbOnStoppe:=Vrai
         ` N'oubliez pas cet appel sinon 4D traitera aussi cet événement
      FILTRER EVENEMENT
   Fin de si
Fin de si
```

Notez que APPELER SUR EVENEMENT est utilisé dans cet exemple car un état spécial est imprimé à l'aide des commandes PARAMETRES IMPRESSION, IMPRIMER LIGNES et SAUT DE PAGE dans une structure de type Boucle..Fin de boucle.

Lorsque vous imprimez un état à l'aide la commande IMPRIMER SELECTION, vous n'avez pas besoin de gérer les événements permettant à l'utilisateur d'interrompre l'impression, IMPRIMER SELECTION le fait pour vous.

Référence

FILTRER EVENEMENT, Majuscule enfoncee, POSITION SOURIS.

FILTRER EVENEMENT

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

FILTRER EVENEMENT ne peut être appelée qu'à l'intérieur d'une méthode de gestion d'événements installée par APPELER SUR EVENEMENT.

Lorsqu'une méthode de gestion d'événements appelle la commande FILTRER EVENEMENT, l'événement courant n'est pas passé à 4e Dimension.

Cette commande vous permet d'effacer l'événement courant (i.e. clic, frappe clavier) de la séquence d'événements, de manière à ce que 4D n'effectue pas de traitement sur l'événement que vous provoquez dans la méthode de gestion d'événements.

ATTENTION : Evitez de créer une méthode de gestion d'événement appelant uniquement FILTRER EVENEMENT car TOUS les événements vont être ignorés par 4D. Si vous vous retrouvez dans un tel cas, vous pouvez sortir de la méthode en tapant **Ctrl**+**Maj**+**Retour Arrière** (sous Windows) ou **Commande**+**Option**+**Maj**+**Contrôle**+**Retour Arrière** (sous MacOS). Dans ce cas, le process de gestion d'événement est converti en process normal n'interceptant plus aucun événement.

Exemples

Référez-vous à l'exemple d'APPELER SUR EVENEMENT.

Référence

APPELER SUR EVENEMENT.

APPELER SUR ERREUR (méthodErreur)

Paramètre	Type		Description
méthodErreur	Alpha	→	Méthode de gestion d'erreur à appeler ou Chaîne vide pour désinstaller la méthode

Description

APPELER SUR ERREUR installe la méthode projet dont le nom est passé dans méthodErreur comme méthode d'interception des erreurs — aussi appelée méthode de gestion des erreurs.

La portée de cette commande est le process courant. Il ne peut y avoir qu'une seule méthode de gestion des erreurs par process, mais il peut exister différentes méthodes de gestions d'erreurs pour plusieurs process.

Pour désinstaller une méthode de gestion des erreurs, appelez de nouveau APPELER SUR ERREUR et passez une chaîne vide dans méthodErreur.

Après l'installation, 4e Dimension appelle cette méthode lorsqu'une erreur se produit.

Vous pouvez identifier les erreurs en lisant la variable système Error, qui contient le code de l'erreur. Les codes d'erreurs retournés par 4e Dimension sont traités dans les sections Codes d'erreurs. Reportez-vous par exemple à la section Erreurs de syntaxe ou Erreurs de la base de données. La variable Error n'est définie qu'à l'intérieur de la méthode de gestion des erreurs ; si vous souhaitez que le code soit accessible dans la méthode ayant provoqué l'erreur, copiez la variable Error dans votre propre variable process.

La méthode de gestion des erreurs doit généralement traiter les erreurs de manière appropriée ou afficher un message d'erreur à l'utilisateur. Les erreurs peuvent être générées par :
• Le moteur de base de données de 4e Dimension ; par exemple, lorsque la sauvegarde d'un enregistrement provoquerait la duplication d'une clé d'index unique.
• L'environnement de 4e Dimension ; par exemple, lorsque vous n'avez pas assez de mémoire pour remplir un tableau.
• Le système d'exploitation sur lequel la base est lancée ; par exemple, disque plein ou erreurs d'entrée/sortie.

La commande STOP peut être utilisée pour stopper le traitement. Si vous n'appelez pas STOP dans la méthode installée, 4e Dimension retourne à la méthode interrompue et reprend son exécution. Utilisez la commande STOP lorsque l'exécution ne peut se poursuivre.

Si une erreur se produit dans la méthode de gestion d'erreurs elle-même, 4e Dimension reprend le contrôle de la gestion des erreurs. En conséquence, assurez-vous que la méthode de gestion des erreurs installée ne puisse pas elle-même générer d'erreur. Aussi, vous ne pouvez pas utiliser la commande APPELER SUR ERREUR dans une méthode de gestion des erreurs.

APPELER SUR ERREUR est généralement placée dans la méthode base d'ouverture d'une base en menus créés, afin de gérer les erreurs pour cette application. APPELER SUR ERREUR peut également être placée au début d'une méthode pour gérer les erreurs spécifiques à cette méthode.

Lorsqu'une méthode APPELER SUR ERREUR est installée, il n'est plus possible de tracer l'exécution des méthodes à l'aide de la combinaison **Alt**+**clic** (sous Windows) ou **Option**+**clic** (sous MacOS). En effet, cette combinaison génère un code d'erreur (code 1006) qui active immédiatement la méthode d'appel sur erreur. Cependant, vous pouvez tester ce code d'erreur et appeler la commande TRACE si nécessaire.

Exemples

(1) La méthode projet suivante tente de créer un document dont le nom est reçu en paramètre et retourne 0 (zéro) ou un code d'erreur si le document n'a pas pu être créé :

```
   ` Méthode projet Créer doc
   ` Créer doc ( Chaîne ; Pointeur ) -> Entier long
   ` Créer doc ( NomDoc ; ->DocRef ) -> Code d'erreur résultant

gError:=0
⇒  APPELER SUR ERREUR("IO TRAITEMENT ERREURS")
   $2->:=Creer document($1)
⇒  APPELER SUR ERREUR("")
   $0:=gError
```

La méthode projet IO TRAITEMENT ERREURS est la suivante :

```
   ` Méthode projet IO TRAITEMENT ERREURS
gError:=Error  ` Simple copie du code d'erreur dans la variable process gError
```

Notez l'utilisation de la variable process gError pour récupérer le code d'erreur dans la méthode en train de s'exécuter. Une fois que ces méthodes sont présentes dans votre base, vous pouvez écrire par exemple :

```
   ` ...
C_HEURE(vhDocRef)
$vlErrCode:=Créer doc($vsDocumentNom;->vhDocRef)
Si ($vlErrCode=0)
      `...
   FERMER DOCUMENT($vlErrCode)
Sinon
   ALERTE ("Le document n'a pas pu être créé, erreur d'E/S "+Chaîne($vlErrCode))
Fin de si
```

(2) Reportez-vous à l'exemple de la section Tableaux et mémoire.

(3) Alors que vous implémentez un ensemble complexe d'opérations, vous pouvez terminer avec de multiples sous-routines qui nécessitent différentes méthodes de gestion des erreurs. Comme ne pouvez avoir qu'une seule méthode à la fois de gestion des erreurs par process, vous devez soit repérer la méthode courante à chaque fois que vous appelez APPELER SUR ERREUR, soit utiliser une variable tableau process (ici tabErrorMethod) pour “empiler” les méthodes de gestion d'erreur ainsi qu'une méthode projet (ici APPEL SUR ERR) pour les installer et les désinstaller. Le tableau doit être initialisé au tout début de l'exécution du process :

```
   ` N'oubliez pas d'initialiser le tableau au début
   ` de la méthode de gestion du process
TABLEAU ALPHA(63;tabErrorMethod;0)
```

Voici la méthode personnalisée APPEL SUR ERR :

```
   ` Méthode projet APPEL SUR ERR
   ` APPEL SUR ERR { ( Chaîne ) }
   ` APPEL SUR ERR { ( Nom de la méthode ) }

C_ALPHA(63;$1;$ErrorMethod)
C_ENTIER LONG($vlElem)

Si (Nombre de parametres>0)
   $ErrorMethod:=$1
Sinon
   $ErrorMethod:=""
Fin de si
```

```
   Si ($ErrorMethod#"")
      C_ENTIER LONG(gError)
      gError:=0
      $vlElem:=1+Taille tableau(tabErrorMethod)
      INSERER LIGNES(tabErrorMethod;$vlElem)
      tabErrorMethod{$vlElem}:=$1
⇒     APPELER SUR ERREUR($1)
   Sinon
⇒     APPELER SUR ERREUR("")
      $vlElem:=Taille tableau(tabErrorMethod)
      Si ($vlElem>0)
         SUPPRIMER LIGNES(tabErrorMethod;$vlElem)
         Si ($vlElem>1)
            APPELER SUR ERREUR(tabErrorMethod{$vlElem-1})
         Fin de si
      Fin de si
   Fin de si
```

Vous pouvez alors l'appeler de la manière suivante :

```
gError:=0
APPEL SUR ERR("ERREURS ES") ` Installe la méthode de gestion d'erreurs ERREURS ES
   ` ...
APPEL SUR ERR("TOUTES ERREURS")
   ` Installe la méthode de gestion d'erreurs TOUTES ERREURS
   ` ...
APPEL SUR ERR
   ` Désinstalle la méthode de gestion d'erreurs TOUTES ERREURS et
   ` réinstalle ERREURS ES
   ` ...
APPEL SUR ERR   ` Désinstalle la méthode de gestion d'erreurs ERREURS ES
   ` ...
```

(4) La méthode de gestion d'erreurs suivante ignore les interruptions de l'utilisateur :

```
   ` Méthode projet MONTRER ERREURS SEULEMENT
Si (Error#1006)
   ALERTE ("L'erreur "+Chaine(Error)+" s'est produite.")
Fin de si
```

Référence

STOP.

Note : Vous aurez rarement besoin de cette commande.

STOP

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

La commande STOP est destinée à être utilisée dans une méthode projet de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Si vous n'avez pas installé de méthode projet de gestion d'erreurs, lorsqu'une erreur se produit (par exemple une erreur de la base de données), 4D affiche sa boîte de dialogue d'erreur standard et interrompt l'exécution de votre code :
• Si le code en cours d'exécution est une méthode objet ou formulaire (ou une méthode projet appelée depuis une méthode formulaire ou objet), 4D "rend la main" au formulaire actuellement affiché.
• Si le code en cours d'exécution est une méthode appelée depuis un menu, 4D "rend la main" à la barre de menus ou au formulaire actuellement affiché.
• Si le code en cours d'exécution est la méthode de gestion d'un process, le process est tué.
• Si le code en cours d'exécution est une méthode appelée directement ou indirectement par une opération d'import ou d'export, cette dernière est stoppée. Il en va de même pour les recherches séquentielles et les tris.
• Etc...

Si vous décidez de traiter les erreurs à l'aide d'une méthode projet d'interception d'erreurs, 4D n'affiche plus sa boîte de dialogue d'erreur standard et n'interrompt plus l'exécution de votre code. Au lieu de cela, 4D appelle votre méthode projet d'interception d'erreurs puis poursuit l'exécution de la ligne de code suivant celle ayant provoqué l'erreur. Vous pouvez traiter certaines erreurs par programmation (par exemple pendant un import, si vous interceptez une erreur de la base de donnée signalant une valeur dupliquée, vous pouvez ignorer l'erreur et poursuivre l'opération). Il existe également des erreurs que vous ne pouvez pas traiter ou des erreurs que vous ne devez pas "ignorer". Dans ces cas, vous devez stopper l'exécution de la méthode comme le fait 4D ; pour cela, appelez la commande STOP depuis la méthode projet d'interception d'erreurs.

Un peu d'histoire...
Bien que la commande STOP soit destinée à une utilisation au sein d'une méthode projet d'interception d'erreurs, des membres de la communauté 4D ont commencé à l'utiliser dans d'autres méthodes projet pour interrompre leur exécution. Le fait que cela fonctionne n'est qu'un "effet secondaire". Nous vous recommandons de n'utiliser cette commande que dans des méthodes projet d'interception d'erreurs.

Référence
APPELER SUR ERREUR.

30

Langage

Nombre de parametres

Langage

version 5

Nombre de parametres → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Nombre de paramètres effectivement passés

Description

Nombre de parametres retourne le nombre de paramètres passés à une méthode projet.

ATTENTION : Nombre de parametres n'a d'intérêt que dans une méthode projet appelée par une autre méthode (projet ou non). Si la méthode projet qui appelle Nombre de parametres est associée à une commande de menu, la fonction retourne 0.

Exemples

(1) Les méthodes projet de 4e Dimension acceptent que des paramètres soient optionnels, à partir de la droite. Par exemple, la méthode maMéthode(a;b;c;d) peut accepter les syntaxes suivantes :

```
maMéthode ( a ; b ; c ; d )  ` Tous les paramètres sont passés
maMéthode ( a ; b ; c )  ` Le dernier paramètre n'est pas passé
maMéthode ( a ; b )  ` Les deux derniers paramètres ne sont pas passés
maMéthode ( a )  ` Seul le premier paramètre est passé
maMéthode  ` Aucun paramètre n'est passé
```

Si vous utilisez Nombre de parametres dans maMéthode, vous pouvez détecter le nombre de paramètres passés et effectuer des opérations différentes selon ce nombre. L'exemple suivant affiche un texte et peut soit l'insérer dans une zone de 4D Write, soit l'écrire dans un document sur disque :

```
` Méthode AJOUTER TEXTE
` AJOUTER TEXTE ( Texte { ; Entier long { ; Heure } } )
` AJOUTER TEXTE ( Texte { ; zone 4D Write { ; RéfDoc } } )

C_TEXTE ($1)
C_HEURE ($2)
C_ENTIER LONG ($3)

MESSAGE ($1)
```

```
⇒       Si (Nombre de parametres>=3)
           ENVOYER PAQUET ($3;$1)
        Sinon
⇒          Si (Nombre de parametres>=2)
              wr_INSERER TEXTE ($2;$1)
           Fin de si
        Fin de si
```

Vous pouvez ensuite appeler cette méthode de ces trois façons différentes :

```
AJOUTER TEXTE (vtTexte)  ` Afficher seulement le message texte
   ` Afficher le message texte et ajouter le texte à $wrZone
AJOUTER TEXTE (vtTexte;$wrZone)
   ` Afficher le message texte et l'écrire dans $vhRéfDoc
AJOUTER TEXTE (vtTexte;0;$vhRéfDoc)
```

(2) Les méthodes projet de 4e Dimension acceptent un nombre variable de paramètres du même type à partir de la droite. Pour déclarer ces paramètres, vous devez utiliser des directives de compilation auxquelles vous passez ${N} en tant que variable, où N spécifie le premier des paramètres. A l'aide de Nombre de parametres, vous pouvez référencer ces paramètres dans une boucle avec la syntaxe d'indirection de paramètre. L'exemple suivant est une fonction qui retourne la valeur maximale reçue en tant que paramètre :

```
   ` Méthode projet Max de
   ` Max de ( Réel { ; Réel2... ; RéelN } ) -> Réel
   ` Max de ( Valeur { ; Valeur2... ; ValeurN } ) -> Valeur maximale

C_REEL ($0;${1})  ` Tous les paramètres et le résultat de la fonction sont de type REEL
$0:=${1}
⇒  Boucle ($vlParam;2;Nombre de parametres)
      Si (${$vlParam}>$0)
         $0:=${$vlParam}
      Fin de si
   Fin de boucle
```

Vous pouvez alors appeler cette méthode d'une des deux manières suivantes :

```
vrRésultat:=Max de (Enregistrements dans ensemble("Opération A");
                                    Enregistrements dans ensemble("Opération B"))
```

ou :

```
vrRésultat:=Max de (r1;r2;r3;r4;r5;r6)
```

Référence

Commandes du thème Compilateur, C_ALPHA, C_BLOB, C_BOOLEEN, C_DATE, C_ENTIER, C_ENTIER LONG, C_GRAPHE, C_HEURE, C_IMAGE, C_POINTEUR, C_REEL, C_TEXTE.

Type (champVar) → Numérique

Paramètre	Type		Description
champVar	Champ | Variable	→	Champ ou variable à tester
Résultat	Numérique	←	Numéro du type de données

Description

Type retourne une valeur numérique qui indique le type du champ ou de la variable que vous avez passé(e) dans le paramètre champVar.

4e Dimension fournit les constantes prédéfinies suivantes :

Constante	Type	Valeur
Est un champ alpha	Entier long	0
Est une variable chaîne	Entier long	24
Est un texte	Entier long	2
Est un numérique	Entier long	1
Est un entier	Entier long	8
Est un entier long	Entier long	9
Est une date	Entier long	4
Est une heure	Entier long	11
Est un booléen	Entier long	6
Est une image	Entier long	3
Est une sous-table	Entier long	7
Est un BLOB	Entier long	30
Est une variable indéfinie	Entier long	5
Est un pointeur	Entier long	23
Est un tableau chaîne	Entier long	21
Est un tableau texte	Entier long	18
Est un tableau numérique	Entier long	14
Est un tableau entier	Entier long	15
Est un tableau entier long	Entier long	16
Est un tableau date	Entier long	17
Est un tableau booléen	Entier long	22
Est un tableau image	Entier long	19
Est un tableau pointeur	Entier long	20
Est un tableau 2D	Entier long	13

Note de compatibilité : Dans les versions précédentes de 4D, la fonction Type, lorsqu'elle était appliquée à une variable de type Graphe déclarée à l'aide de la commande C_GRAPHE, retournait 3 (Est une image). A compter de la version 6 de 4D, Type retourne 9 (Est un entier long) lorsqu'elle est appliquée à une variable de type Graphe.

Vous pouvez appliquer la fonction Type aux champs, variables interprocess, variables process, variables locales et à des pointeurs dépointés qui réfèrent ces types d'objets.

Note version 6 : A partir de la version 6 de 4D, vous pouvez appliquer Type aux paramètres ($1,$2..., ${...}) d'une méthode projet ou au résultat d'une fonction ($0).

Exemples

(1) Référez-vous à l'exemple de la commande AJOUTER A PRESSE PAPIERS.

(2) Référez-vous à l'exemple de la commande PROPRIETES GLISSER DEPOSER.

(3) La méthode projet suivante efface une partie ou la totalité des champs de l'enregistrement courant de la table vers laquelle pointe le pointeur passé en paramètre, et ce, sans supprimer l'enregistrement ou changer d'enregistrement courant :

```
      ` Méthode projet VIDER ENREGISTREMENT
      ` VIDER ENREGISTREMENT ( Pointeur {; Entier long } )
      ` VIDER ENREGISTREMENT ( -> [Table] { ; Type des valeurs } )

   C_POINTEUR ($1)
   C_ENTIER LONG ($2;$vlTypeVal)

   Si (Nombre de parametres>=2)
      $vlTypeVal:=$2
   Sinon
      $vlTypeVal:=0xFFFFFFFF
   Fin de si
   Boucle ($vlChamp;1;Nombre de champs($1))
      $vpChamp:=Champ(Table($1);$vlChamp)
⇒     $vlTypeChamp:=Type($vpChamp->)
      Si ( $vlTypeVal ?? $vlTypeChamp )
         Au cas ou
            : (($vlTypeChamp=Est un champ alpha)|($vlTypeChamp=Est un texte))
               $vpChamp->:=""
            : (($vlTypeChamp=Est un numérique)|($vlTypeChamp=Est un entier)|
                                                ($vlTypeChamp=Est un entier long))
               $vpChamp->:=0
            : ($vlTypeChamp=Est une date)
               $vpChamp->:=!00/00/00!
```

```
            : ($vlTypeChamp=Est une heure)
               $vpChamp->:=?00:00:00?
            : ($vlTypeChamp=Est un booléen)
               $vpChamp->:=Faux
            : ($vlTypeChamp=Est une image)
               C_IMAGE($vgImageVide)
               $vpChamp->:=$vgImageVide
            : ($vlTypeChamp=Est une sous-table)
               Repeter
                  TOUS LES SOUS ENREGISTREMENTS($vpChamp->)
                  SUPPRIMER SOUS ENREGISTREMENT($vpChamp->)
               Jusque(Sous enregistrements trouves($vpChamp->)=0)
            : ($vlTypeChamp=Is BLOB)
               FIXER TAILLE BLOB($vpChamp->;0)
         Fin de cas
      Fin de si
   Fin de boucle
```

Une fois cette méthode projet implémentée dans votre base, vous pouvez écrire :

```
   ` Effacer la totalité du contenu de l'enregistrement courant de la
   ` table [Choses à faire]
VIDER ENREGISTREMENT (->[Choses à faire])

   ` Effacer les champs de type Texte, BLOB et Image de l'enregistrement courant de la
   ` table [Choses à faire]
VIDER ENREGISTREMENT (->[Choses à faire]; 0 ?+ Est un texte ?+ Est un BLOB ?
                                                    + Est une image )

   ` Effacer la totalité de l'enregistrement courant de la table [Choses à faire] sauf les
   ` champs Alpha
VIDER ENREGISTREMENT (->[Choses à faire]; -1 ?- Est un champ alpha )
```

Référence

Est une variable, Indefinie.

Self → Pointeur

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Pointeur	←	Pointeur vers l'objet du formulaire dont la méthode est en cours d'exécution (le cas échéant), sinon Nil (->[]) si hors contexte

Description

Self retourne un pointeur vers l'objet du formulaire dont la méthode objet est en cours d'exécution.

La fonction Self est utilisée pour référencer une variable dans sa propre méthode objet. Elle ne retourne un pointeur valide que si elle est appelée dans une méthode objet ; elle ne peut donc pas être utilisée dans une méthode projet, même appelée par une méthode objet. Si Self est appelée en-dehors de ce contexte, elle retourne un pointeur Nil (->[]).

Conseil : Self est très utile lorsque plusieurs objets d'un formulaire doivent effectuer la même action, opérée sur eux-mêmes.

Exemple

Référez-vous à l'exemple de la commande RESOUDRE POINTEUR.

Référence

RESOUDRE POINTEUR.

RESOUDRE POINTEUR (pointeur; nomVar; numTable; numChamp)

Paramètre	Type		Description
pointeur	Pointeur	→	Pointeur duquel récupérer l'objet référencé
nomVar	Alpha	←	Nom de la variable référencée ou chaîne vide
numTable	Numérique	←	Numéro de la table ou de l'élément de tableau référencé(e) ou 0 ou -1
numChamp	Numérique	←	Numéro du champ référencé ou 0

Description

RESOUDRE POINTEUR récupère l'information de l'objet référencé par pointeur et la retourne dans les paramètres nomVar, numTable et numChamp.

Selon la nature de l'objet référencé par le pointeur, RESOUDRE POINTEUR retourne les valeurs suivantes :

Objet référencé	Paramètres		
	nomVar	**numTable**	**numChamp**
Aucun (pointeur NIL)	"" (chaîne vide)	0	0
Variable	Nom de la variable	-1	0
Tableau	Nom du tableau	-1	0
Elément de tableau	Nom du tableau	numéro de l'élément	0
Table	"" (chaîne vide)	numéro de la table	0
Champ	"" (chaîne vide)	numéro de la table	numéro du champ

Note : Si la valeur que vous passez dans le paramètre pointeur n'est pas de type pointeur, une erreur de syntaxe est générée.

Exemples

(1) Dans un formulaire, vous créez un groupe de 100 variables saisissables qui s'appellent v1, v2... v100. Pour cela, vous procédez de la manière suivante :
• Vous créez une variable saisissable que vous appelez v.
• Vous définissez les propriétés de l'objet suivant vos besoins.
• Vous associez la méthode suivante à l'objet :

```
FaireQuelqueChose (Self) ` FaireQuelqueChose est une méthode projet de la base
```

• Vous pouvez alors soit dupliquer la variable autant de fois que nécessaire, soit utiliser la fonctionnalité Tableau sur la grille de l'éditeur de formulaires.
• Dans la méthode FaireQuelqueChose, si vous voulez connaître l'indice de la variable pour laquelle la méthode est appelée, vous écrivez le code suivant :

```
⇒ RESOUDRE POINTEUR($1;$vaNomVar;$vlNumTable;$vlNumChamp)
  $vlVarNum:=Num(Sous chaine($vaNomVar;2))
```

• En suivant ces étapes, vous avez écrit une fois seulement les méthodes objet pour les 100 variables : vous n'avez pas eu besoin d'écrire FaireQuelqueChose(1), FaireQuelqueChose(2)..., FaireQuelqueChose(100)).

(2) Pour des raisons de débogage, vous voulez vérifier si le deuxième paramètre ($2) d'une méthode est un pointeur vers une table. Le début de votre méthode peut être écrit ainsi :

```
      ` ...
   Si (◊Débogage)
⇒     RESOUDRE POINTEUR($2;$vaNomVar;$vlNumTable;$vlNumChamp)
      Si (Non(($vlNumTable>0)&($vlNumChamp=0)&($vlNomVar="")))
         ` ATTENTION : Le pointeur n'est pas une référence à une table
         TRACE
      Fin de si
   Fin de si
      ` ...
```

(3) Reportez-vous à l'exemple de la commande PROPRIETES GLISSER DEPOSER

Référence

Champ, Est une variable, Nil, Pointeur vers, PROPRIETES GLISSER DEPOSER, Table.

Nil (unPointeur) → Booléen

Paramètre	Type		Description
unPointeur	Pointeur	→	Pointeur à tester
Résultat	Booléen	←	VRAI = Pointeur Nil (->[]) FAUX = Pointeur valide vers un objet existant

Description

Nil retourne Vrai si le pointeur que vous passez dans unPointeur est Nil (->[]). Elle retourne Faux dans tous les autres cas (pointeur vers un champ, une table ou une variable).

A compter de la version 6 de 4e Dimension, vous pouvez utiliser la commande RESOUDRE POINTEUR qui vous indique le type de l'objet référencé, quel qu'il soit (pointeurs Nil compris).

Référence

Est une variable, RESOUDRE POINTEUR.

Est une variable

Est une variable (unPointeur) → Booléen

Paramètre	Type		Description
unPointeur	Pointeur	→	Pointeur à tester
Résultat	Booléen	←	VRAI = Pointeur pointe vers une variable FAUX = Pointeur ne pointe pas vers une variable

Description

La fonction Est une variable retourne Vrai si le pointeur passé dans le paramètre unPointeur référence une variable définie. Elle retourne Faux dans tous les autres cas (pointeur vers un champ ou table, pointeur Nil, etc.).

A partir de la version 6, vous pouvez utiliser la commande RESOUDRE POINTEUR qui vous indique la nature de l'objet référencé, quel qu'il soit (pointeurs Nil compris).

Référence

Nil, RESOUDRE POINTEUR.

Pointeur vers (nomVar) → Pointeur

Paramètre	Type		Description
nomVar	Alpha	→	Nom d'une variable process
Résultat	Pointeur	←	Pointeur vers une variable process

Description

Pointeur vers retourne un pointeur vers la variable dont le nom est passé dans nomVar.

Pour récupérer un pointeur vers un champ, utilisez la fonction Champ. Pour récupérer un pointeur vers une table, utilisez la fonction Table.

Note : Vous devez passer à LIRE VARIABLES un *nom de variable* uniquement. Il n'est pas possible d'utiliser des *expressions*, telles que, par exemple, $tTabNom+"{3}".

Exemple

Dans un formulaire, vous construisez une grille de 5 X 10 variables saisissables dont les noms sont v1, v2... v50. Pour initialiser toutes ces variables, vous pouvez écrire :

```
      ` ...
   Boucle ($vlVar;1;50)
⇒     $vpVar:=Pointeur vers("v"+Chaine($vlVar))
      $vpVar->:=""
   Fin de boucle
```

Référence

Champ, Table.

EXECUTER (instruction)

Paramètre	Type		Description
instruction	Alpha	→	Code à exécuter

Description

EXECUTER exécute instruction comme une ligne de code. Cette chaîne d'instructions doit comporter une seule ligne. Si instruction est une chaîne vide, EXECUTER ne fait rien.

Le principe est que si instruction peut être exécutée comme une méthode d'une seule ligne, alors elle s'exécutera correctement. La commande EXECUTER doit être utilisée avec précautions, car elle ralentit la vitesse d'exécution. Dans une base compilée, le code d'instruction n'est pas compilé. Cela signifie que l'instruction sera bien exécutée, mais ne sera pas vérifiée par 4D Compiler au moment de la compilation.

L'instruction peut contenir les éléments suivants :
• un appel à une méthode projet,
• un appel à une commande 4D,
• une assignation.

La formule peut utiliser des variables process et interprocess. En revanche, instruction ne doit pas contenir d'instructions de contrôle de flux (Si, Tant que...) car le code doit "tenir" sur une seule ligne .

Exemples

Reportez-vous à l'exemple de la commande Nom commande.

Référence

Nom commande.

Nom commande (commande) → Alpha

Paramètre	Type		Description
commande	Numérique	→	Numéro de la commande
Résultat	Alpha	←	Nom (traduit) de la commande

Description

La fonction Nom commande retourne le nom (universel) de la commande dont le numéro a été passé dans commande.

4e Dimension comporte un système unique en son genre de traduction dynamique des mots-clés, constantes et noms de commandes que vous utilisez dans vos méthodes. Si, par exemple, vous écrivez dans la version anglaise de 4D :

```
DEFAULT TABLE ([MyTable])
ALL RECORDS ([MyTable])
```

Le même code, si vous l'ouvrez avec la version française de 4D, sera automatiquement traduit :

```
TABLE PAR DEFAUT ([MyTable])
TOUT SELECTIONNER ([MyTable])
```

Cependant, 4e Dimension comporte aussi une autre fonctionnalité unique, la commande EXECUTER, qui vous permet de construire à la volée et d'exécuter des parties de code même lorsque la base de données est compilée.

Si nous réécrivons le code précédent avec des commandes EXECUTE, en anglais, il prendra l'apparence suivante :

```
EXECUTE ( "DEFAULT TABLE([MyTable])")
EXECUTE ( "ALL RECORDS([MyTable])")
```

Le même code, ouvert avec la version française de 4D, sera automatiquement traduit :

```
EXECUTER ( "DEFAULT TABLE([MyTable])")
EXECUTER ( "ALL RECORDS([MyTable])")
```

4D traduit automatiquement la commande EXECUTE (anglais) en EXECUTER (français) mais ne peut pas traduire les instructions passées à la commande.

Si vous utilisez la commande EXECUTER dans votre application et si vous souhaitez éliminer les problèmes de traduction liés à ce type d'instructions, utilisez Nom commande pour rendre vos instructions indépendantes des localisations. L'exemple précédent devient alors :

```
⇒    EXECUTE (Command name (46)+"([MyTable])")
⇒    EXECUTE (Command name (47)+"([MyTable])")
```

La version française se lira :

```
⇒    EXECUTER (Nom commande (46)+"([MyTable])")
⇒    EXECUTER (Nom commande (47)+"([MyTable])")
```

Note : Pour connaître le numéro d'une commande, sélectionnez-la dans la fenêtre de l'Explorateur. Son numéro apparaît alors dans la partie droite de l'Explorateur.

Exemples

(1) Pour toutes les tables de votre base de données, vous avez créé un formulaire appelé "FORMULAIRE ENTREE" que vous utilisez pour la saisie dans chaque table. Vous voulez ajouter une méthode projet générique qui va désigner ce formulaire comme étant le formulaire entrée pour la table dont vous passez le nom ou le pointeur. Vous pouvez écrire :

```
   ` méthode projet FORMULAIRE ENTREE STANDARD
   ` FORMULAIRE ENTREE STANDARD ( Pointeur {; Chaîne })
   ` FORMULAIRE ENTREE STANDARD ( ->Table {; NomTable })
C_POINTEUR ($1)
C_ALPHA (31;$2)

Si (Nombre de parametres>=2)
⇒    EXECUTER (Nom commande (55)+"(["+$2+"];"FORMULAIRE ENTREE")")
Sinon
   Si (Nombre de parametres>=1)
      FORMULAIRE ENTREE ($1->;"FORMULAIRE ENTREE")
   Fin de si
Fin de si
```

Une fois que cette méthode a été ajoutée dans votre base, vous pouvez écrire :

```
FORMULAIRE ENTREE STANDARD  (->[Employés])
FORMULAIRE ENTREE STANDARD  ("Employés")
```

Note : Généralement, il est préférable d'utiliser des pointeurs pour écrire des routines génériques. Tout d'abord, le code sera exécuté compilé si la base est compilée. Deuxièmement, 4D Insider récupérera les références des objets vers lesquels vous passez les pointeurs. Troisièmement, dans l'exemple ci-dessus, votre code cessera de fonctionner si vous renommez la table. Cependant, dans certains cas, l'utilisation de la commande EXECUTER peut être la réponse à vos besoins.

(2) Dans un formulaire, vous voulez afficher une liste déroulante contenant les commandes standard de génération d'états. Dans la méthode objet de cette liste déroulante, vous écrivez :

```
      Au cas ou
         : (Evenement formulaire =Sur Chargement)
            TABLEAU TEXTE (asCommand;4)
⇒           asCommand{1}:=Nom commande (1) ` Somme
⇒           asCommand{2}:=Nom commande (2) ` Moyenne
⇒           asCommand{3}:=Nom commande (3) ` Min
⇒           asCommand{4}:=Nom commande (4) ` Max
         ` ...
      Fin de cas
```

Dans une version anglaise de 4D, la liste déroulante contiendra : Sum, Average, Min et Max.
Dans une version française de 4D, la liste déroulante contiendra : Somme, Moyenne, Min et Max.

Référence

EXECUTER.

Nom methode courante

Nom methode courante → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Nom de la méthode d'appel

Description

La commande Nom methode courante retourne le nom de la méthode dans laquelle elle est appelée. Cette commande est utile dans le cadre du débogage de méthodes génériques.

En fonction du type de méthode d'appel, la chaîne retournée peut prendre l'une des formes suivantes :

Méthode d'appel	Chaîne retournée
Méthode base	NomMéthode
Trigger	Trigger sur [NomTable]
Méthode projet	NomMéthode
Méthode formulaire	[NomTable]NomFormulaire
Méthode objet	[NomTable]NomFormulaire.NomObjet

Cette commande ne doit pas être appelée depuis une formule 4D.

TRACE

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

La commande TRACE est utilisée, lors du développement des bases, pour tracer des méthodes, c'est-à-dire contrôler leur exécution pas à pas.

La commande TRACE affiche le débogueur de 4e Dimension dans le process courant. La fenêtre du débogueur apparaît dès que la commande est appelée, avant l'exécution de la ligne de code suivante, et reste affichée pour l'exécution de chaque ligne de code. Vous pouvez également appeler manuellement le débogueur en utilisant la combinaison **Alt**+**clic** sous Windows ou **Option**+**clic** sous MacOS pendant l'exécution du code.

Dans les bases compilées, cette commande est ignorée.

4D Server : Si vous appelez TRACE depuis une méthode projet exécutée en tant que Procédure stockée, la fenêtre du débogueur apparaîtra sur le poste serveur.

Conseils

(1) N'appelez pas TRACE lorsque vous utilisez un formulaire pour lequel les événements Sur activation et Sur désactivation ont été sélectionnés. En effet, chaque fois que la fenêtre du débogueur apparaîtra, ils seront activés, et cela créera une boucle sans fin entre les événements et le débogueur. Si vous vous retrouvez dans cette situation, pour en sortir, passez en mode Structure en cliquant sur une des fenêtres de ce mode, puis :
• Si l'appel à TRACE se trouve dans une méthode projet ou objet (méthodes rechargées à chaque exécution), supprimez-la. Cela stoppera la boucle sans fin.
• Si l'appel à TRACE se trouve dans la méthode du formulaire (qui n'est pas rechargée tant que vous ne quittez pas le formulaire — ce que vous ne pouvez pas faire à cause de la boucle sans fin), vous pouvez soit quitter et réouvrir la base, soit tuer le process concerné. Si le process ne peut pas être tué, vous devrez adopter la première solution.

(2) Si vous appelez la commande TRACE depuis une méthode formulaire ou objet exécutée pendant la mise à jour du formulaire à l'écran, vous devrez également faire face à un problème de répétition sans fin de la séquence mises à jour du formulaire/apparitions de la fenêtre du débogueur. Dans ce cas, appuyez sur les touches **Alt** + **Maj** (sous Windows) ou **Option**+**Maj** (sous MacOS).

Cela désactivera les événements de mise à jour de la fenêtre courante et, par conséquent, arrêtera l'appel à la commande TRACE via les méthodes objet ou formulaire. Vous pourrez alors passer en mode Structure et supprimer l'appel à TRACE.

Notez que ces deux conseils s'appliquent également aux situations identiques générées par la présence de points d'arrêts permanents dans votre code. En ce qui concerne le paragraphe (1), quel que soit l'endroit où est placé le point d'arrêt, vous pouvez l'enlever et ainsi supprimer les apparitions de la fenêtre de débogage sans devoir quitter et réouvrir la base.

Exemple

Dans le code suivant, la variable process CREER_LANG doit être égale à “US” ou “FR”. Si ce n'est pas le cas, la méthode projet DEBUG est appelée :

```
   ` ...
Au cas ou
   : (CREER_LANG="US")
      vsBHCmdNom:=[Commandes]CM US Nom
   : (CREER_LANG="FR")
      vsBHCmdNom:=[Commandes]CM FR Nom
Sinon
   DEBUG ("Valeur de CREER_LANG incorrecte")
Fin de cas
```

La méthode projet DEBUG est listée ci-dessous :

```
   ` Méthode projet DEBUG
   ` DEBUG (Texte)
   ` DEBUG (Informations supplémentaires de débogage)

C_TEXTE ($1)
Si (<>vbDebugOn) ` Variable interprocess définie dans la méthode base Sur ouverture
   Si (Application compilee)
      Si (Nombre de parametres>=1)
         ALERTE ($1+Caractere(13)+"Appelez le concepteur au 05 05 05 05")
      Fin de si
   Sinon
⇒        TRACE
   Fin de si
Fin de si
```

Référence

PAS DE TRACE.

PAS DE TRACE

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

La commande PAS DE TRACE est utilisée en phase de développement d'une base de données, pour contrôler l'exécution des méthodes.

PAS DE TRACE désactive le débogueur appelé par la commande TRACE, par une erreur ou par l'utilisateur. Utiliser PAS DE TRACE équivaut à cliquer sur le bouton **Pas de trace** dans la fenêtre de déboguage.

Dans les bases compilées, cette commande est ignorée.

Référence

TRACE.

31

Liens

Présentation des liens

Les commandes de ce thème, en particulier CHARGER SUR LIEN et LIEN RETOUR, établissent et gèrent les relations entre les tables, à la fois pour les liens automatiques et les liens manuels. Consultez le manuel *Mode Structure* de 4e Dimension pour la création des liens entre les tables avant d'utiliser ces commandes.

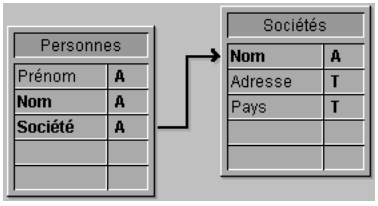
Exploiter par programmation les liens automatiques entre les tables

Deux tables peuvent être reliées par un lien automatique. En général, quand une lien automatique est créé, les enregistrements liés sont chargés et sélectionnés dans la table liée. Un grand nombre d'opérations exploitent cette relation.

En particulier, citons les opérations suivantes :
• Saisie de données,
• Liste des enregistrements à l'écran dans un formulaire de sortie,
• Etats,
• Opérations sur une sélection d'enregistrements, comme les recherches, tris, formules.

Pour améliorer les performances, quand 4D active les liens automatiques, seul un enregistrement devient l'enregistrement courant pour la table. Pour chacune des opérations énumérées ci-dessus, l'enregistrement lié est chargé selon les principes suivants :
• Si un lien sélectionne un seul enregistrement de la table liée, cet enregistrement est chargé du disque.
• Si un lien sélectionne plusieurs enregistrements de la table liée, une nouvelle sélection d'enregistrements est créée pour cette table, et le premier enregistrement de cette sélection est chargé du disque.

Par exemple, dans la structure ci-dessous, si un enregistrement de la table [Personnes] est chargé et affiché pour une saisie de données, l'enregistrement lié dans la [Sociétés] est chargé. De même, si un enregistrement de la table [Sociétés] est chargé et affiché en saisie de données, les enregistrements liés de la table [Personnes] sont sélectionnés.



Dans le type de structure présentée ci-dessus, la table [Personne] est appelée **Table N**, et la table [Société] est appelée **Table 1**. Pour vous souvenir de la différence, pensez à “Beaucoup de personne travaillent dans une société” et “1 société emploie N personnes.”

De même, le champ Société dans la table [Personnes] est appelé **Champ N**, et le champ Nom de la table [Sociétés] est appelé **Champ 1**.

Il n'est pas toujours possible que le champ lié soit unique. Par exemple, le champ [Sociétés]Nom peut contenir des noms de sociétés identiques. Ce cas de non-unicité est facilement contournable : il suffit de créer une relation vers un autre champ de la table liée qui, lui, sera toujours unique. Ce champ pourrait être par exemple un numéro d'identification de la société.

Les commandes listées ci-dessous utilisent les relations automatiques pour charger les enregistrements liés lors de leur exécution. Toutes ces commandes établiront automatiquement un lien "de N vers 1". Seules les commandes signalées explicitement par un Oui établissent automatiquement un lien "de 1 vers N".

Commande	**Lien 1 vers N**
AJOUTER ENREGISTREMENT	Oui
AJOUTER SOUS ENREGISTREMENT	Non
APPLIQUER A SELECTION	Non
VISUALISER SELECTION	Non
ECRITURE DIF	Non
ECRITURE SYLK	Non
ECRITURE ASCII	Non
MODIFIER ENREGISTREMENT	Oui
MODIFIER SOUS ENREGISTREMENT	Non
MODIFIER SELECTION	Oui (en saisie de données)
TRIER	Non
TRIER PAR FORMULE	Non
CHERCHER PAR FORMULE	Oui
CHERCHER DANS SELECTION	Oui
CHERCHER	Oui
IMPRIMER ETIQUETTES	Non
IMPRIMER SELECTION	Oui
ETAT	Non
SELECTION VERS TABLEAU	Non
SELECTION LIMITEE VERS TABLEAU	Non

Activer par programmation les liens entre les tables

Que les relations soient automatiques ne signifie pas que les enregistrements liés ou les enregistrements pour une table seront sélectionnés simplement parce qu'une commande charge un enregistrement. Après avoir exécuté une commande chargeant un enregistrement, dans certains cas vous devez explicitement appeler le ou les enregistrement(s) lié(s) avec CHARGER SUR LIEN ou LIEN RETOUR si vous avez besoin d'accéder aux données liées.

Certaines des commandes listées ci-dessus (par exemple les commandes de recherche) chargent l'enregistrement courant une fois leur tâche terminée. Dans ce cas, l'enregistrement chargé n'appelle pas automatiquement le ou les enregistrement(s) lié(s). Là aussi, vous devez explicitement sélectionner le ou les enregistrement(s) lié(s) avec CHARGER SUR LIEN ou LIEN RETOUR si vous avez besoin d'accéder aux données liées.

Référence

ANCIEN LIEN RETOUR, CHARGER ANCIEN, CHARGER SUR LIEN, CREER SUR LIEN, JOINTURE, LIEN RETOUR, LIENS AUTOMATIQUES, SELECTION RETOUR, STOCKER ANCIEN, STOCKER SUR LIEN.

LIENS AUTOMATIQUES

version 5

LIENS AUTOMATIQUES (lienAppel{; lienRetour})

Paramètre	Type		Description
lienAppel	Booléen	→	Liens N vers 1
lienRetour	Booléen	→	Liens 1 vers N

Description

LIENS AUTOMATIQUES transforme tous les liens manuels en liens automatiques pour toute la base. Cette modification est temporaire et peut à tout moment être remise en cause par un nouvel appel à LIENS AUTOMATIQUES.

Si lienAppel est vrai, tous les liens N vers 1 deviennent automatiques. Si lienAppel est faux, tous les liens N vers 1 deviennent manuels.

Si lienRetour est vrai, tous les liens 1 vers N deviennent automatiques. Si lienRetour est faux, tous les liens 1 vers N deviennent manuels.

Les liens définis comme automatiques en mode Structure ne sont pas affectés par cette commande. Si tous les liens sont définis comme manuels en mode Structure, cette commande vous permet de les rendre automatiques avant d'exécuter des opérations nécessitant qu'ils soient automatiques (par exemple, des recherches et tri relationnels). Après l'exécution de l'opération, le lien peut redevenir manuel.

Exemples

L'exemple suivant rend tous les liens N vers 1 automatiques et rétablit en manuel tous les liens 1 vers N qui étaient précédemment modifiés :

```
⇒ LIENS AUTOMATIQUES (Vrai; Faux)
```

Référence

Présentation des liens, SELECTION VERS TABLEAU, SOUS SELECTION VERS TABLEAU.

CHARGER SUR LIEN (tableN | champN{; discriminant})

Paramètre	Type		Description
tableN | champN	Table | Champ	→	Table pour laquelle définir tous les liens automatiques ou Champ avec lien manuel partant vers la table 1
discriminant	Champ	→	Champ discriminant de la table 1

Description

CHARGER SUR LIEN accepte deux syntaxes.

La première syntaxe de la commande, CHARGER SUR LIEN(tableN), active tous les liens aller automatiques (de N vers 1) pour la table tableN dans le process courant. Cela signifie que pour chaque champ de la tableN d'où part un lien aller automatique, la commande sélectionnera l'enregistrement lié dans chaque table liée. Cela peut donc modifier l'enregistrement courant dans la (les) table(s) liée(s) du process courant.

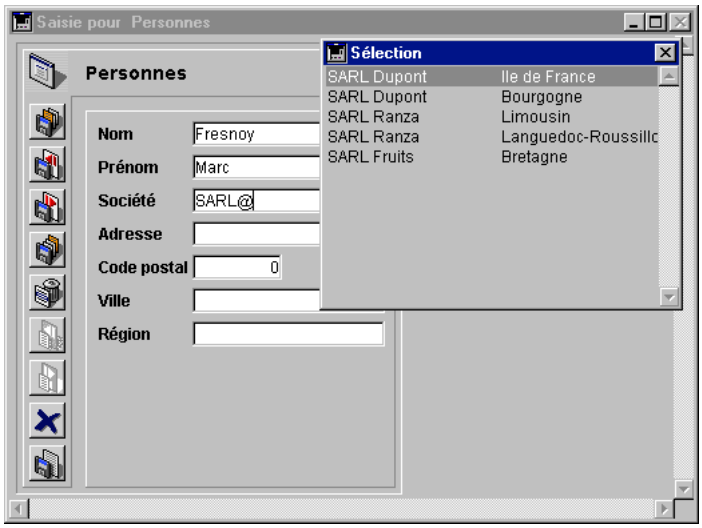
La seconde syntaxe, CHARGER SUR LIEN(champN{;discriminant}), recherche l'enregistrement lié au champ champN. Il n'est pas nécessaire que le lien soit automatique. S'il existe, CHARGER SUR LIEN charge en mémoire l'enregistrement lié, et en fait l'enregistrement et la sélection courants de la table à laquelle il appartient.

Le paramètre optionnel discriminant ne peut être spécifié que si champN est de type Alpha. Le champ discriminant doit être un champ de la table liée. Il peut être de type Alpha, Numérique, Date, Heure ou Booléen. Autrement dit, il ne peut être du type Texte, Image, BLOB ou Sous-table.

Si champN est spécifié et si plus d'un enregistrement est trouvé dans la table liée, CHARGER SUR LIEN affiche une liste des enregistrements qui correspondent à la valeur de champN, permettant à l'utilisateur de sélectionner un enregistrement. Dans cette liste, la colonne de gauche affiche les valeurs des champs liés, la colonne de droite affiche les valeurs de disciminant.

Généralement, plusieurs enregistrements sont trouvés lorsque champN se termine par le caractère Joker (@). S'il n'y en a qu'un seul, la liste de sélection n'apparaît pas.

Dans l'écran ci-dessous, un enregistrement est en train d'être saisi et une liste de sélection s'affiche au premier plan.



La commande suivante a fait apparaître la liste de sélection :

⇒ **CHARGER SUR LIEN** ([Personnes]Société; [Sociétés]Région)

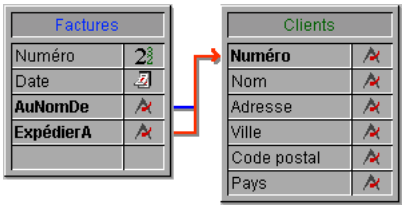
L'utilisateur a saisi SARL@ pour visualiser la liste de toutes les sociétés dont le nom commence par SARL, ainsi que leur région.

Spécifier un champ dans discriminant est la même opération que celle qui consiste à définir un champ discriminant dans la boîte de dialogue de définition des propriétés d'un lien en mode Structure. Pour plus d'informations sur la définition d'un champ discriminant, reportez-vous au manuel Mode Structure de 4e Dimension.

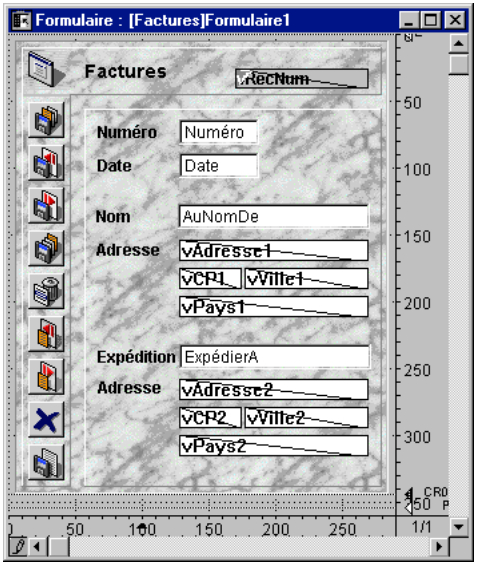
CHARGER SUR LIEN fonctionne avec les liens vers des sous-tables, mais il doit y avoir un lien vers la table parente et vers le champ lié de la sous-table pour que la relation fonctionne correctement. Lorsque vous utilisez une relation vers un sous-enregistrement, vous devez dans un premier temps appeler CHARGER SUR LIEN pour charger en mémoire l'enregistrement lié, puis appeler une seconde fois CHARGER SUR LIEN pour la sous-table.

Exemple

Dans l'exemple suivant, la table [Factures] est reliée à la table [Clients] par deux liens manuels. Un lien part du champ [Factures]AuNomDe et va vers le champ [Clients]Numéro, l'autre lien va de [Factures]ExpédierA à [Clients]Numéro.



Voici le formulaire de la table [Factures] affichant les informations "AuNomDe" et "ExpédierA".



Comme les deux liens pointent vers la même table, [Clients], l'information qu'ils récupèrent doit être affichée dans des variables. Si le formulaire contenait les champs de [Clients], seules les valeurs issues du second lien seraient affichées.

Les deux méthodes suivantes sont les méthodes objet des champs [Factures]ExpédierA et [Factures]AuNomDe. Voici la méthode objet du champ [Factures]AuNomDe :

```
⇒    CHARGER SUR LIEN ([Factures]AuNomDe; [Clients]Adresse)
     vAdress1 := [Clients]Adresse
     vVille1 := [Clients]Ville
     vPays1 := [Clients]Pays
     vCode1 := [Clients]CP
```

Voici la méthode objet du champ [Factures]ExpédierA :

```
⇒    CHARGER SUR LIEN ([Factures]ExpédierA; [Clients]Adresse)
     vAdress2 := [Clients]Adresse
     vVille2 := [Clients]Ville
     vPays2 := [Clients]Pays
     vCode2 := [Clients]CP
```

Référence

CHARGER ANCIEN, LIEN RETOUR.

LIEN RETOUR (table1 | champ1)

Paramètre	Type	Description
table1 | champ1	Table | Champ →	Table pour laquelle établir tous les liens de 1 vers N ou Champ 1

Description

LIEN RETOUR a deux syntaxes.

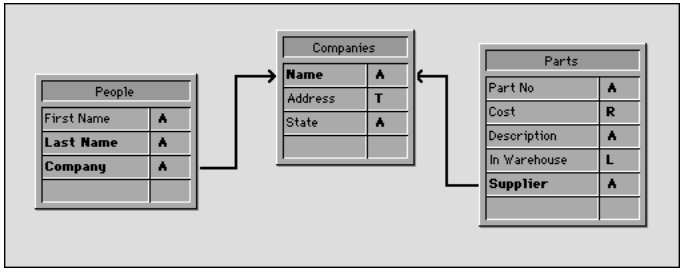
La première syntaxe, LIEN RETOUR (table1), établit tous les liens 1 vers N pour table1. Elle modifie la sélection courante pour chaque table qui a un lien 1 vers N vers table1. Les sélections courantes dans les tables N dépendent de la valeur courante de chaque champ lié dans la table 1. Chaque fois que cette commande est exécutée, les sélections courantes des tables N sont modifiées.

La seconde syntaxe, LIEN RETOUR (champ1), établit le lien 1 vers N pour champ1. Elle modifie la sélection courante pour chaque table qui a un lien avec champ1. En conséquence, les enregistrements liés deviennent la sélection courante de la table N.

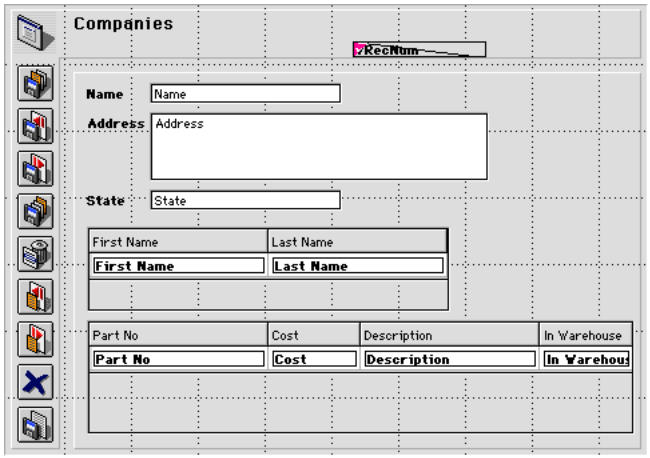
Note : Si la sélection courante de la table 1 est vide au moment de l'exécution de LIRE VARIABLES, la commande ne fait rien.

Exemple

Dans l'exemple suivant, trois tables sont reliées par des liens automatiques. Les deux tables [People] et [Parts] ont un lien N vers 1 vers la table [Companies].



Voici le formulaire pour la table [Companies] qui affiche les enregistrements liés venant des tables [People] et [Parts].



Lorsque le formulaire pour People et Parts s'affiche, les enregistrements liés pour les tables [People] et [Parts] sont chargés et deviennent les sélections courantes de ces tables.

En contrepartie, les enregistrements liés ne sont pas chargés si un enregistrement de la table [Companies] est sélectionné par programmation. Dans ce cas, il faut utiliser la commande LIEN RETOUR.

Par exemple, la méthode suivante effectue une boucle sur chaque enregistrement de la table [Companies]. Pour chaque société, une alerte apparaît. Cette alerte affiche le nombre de personnes dans la société (le nombre d'enregistrements liés dans la table [People]) ainsi que le nombre de Parts que la société distribue (le nombre d'enregistrements dans la table [Parts] qui sont liés). Notez que nous avons besoin d'appeler la commande LIEN RETOUR bien que les liens soient automatiques :

```
    ` Sélectionner tous les enregistrements dans la table
TOUT SELECTIONNER ([Companies])
    ` Trier les enregistrements dans l'ordre alphabétique
TRIER ([Companies]; [Companies]Nom)
    ` Boucler une fois par enregistrement
Boucle ($i; 1; Enregistrements dans table ([Companies]))
⇒       LIEN RETOUR ([Companies]Nom)  ` Sélectionner les enregistrements liés
        ALERTE ("Société : " + [Companies]Nom + Caractere (13) + "personnes dans
                                la société : " + Chaine(Enregistrements trouves ([People]))
                                + Caractere(13) +"Nombre de Produits qu'ils distribuent : "
                                        + Chaine (Enregistrements trouves ([Parts])))
        ENREGISTREMENT SUIVANT ([Companies])  ` Aller à l'enregistrement suivant
Fin de boucle
```

Référence

ANCIEN LIEN RETOUR, CHARGER SUR LIEN.

CREER SUR LIEN (champ)

Paramètre	Type		Description
champ	Champ	→	Champ N (champ d'où part le lien)

Description

CREER SUR LIEN a deux effets. S'il n'existe pas d'enregistrement lié à champ (c'est-à-dire si la valeur courante de champ n'est présente dans le champ correspondant d'aucun enregistrement de la table liée), CREER SUR LIEN crée un nouvel enregistrement lié. Si vous souhaitez conserver dans cet enregistrement la valeur de champ ayant provoqué sa création, assignez-la au champ correspondant. Utilisez ensuite la commande STOCKER SUR LIEN pour sauvegarder le nouvel enregistrement.

Si un enregistrement lié existe déjà, la commande CREER SUR LIEN a alors exactement le même effet que CHARGER SUR LIEN : elle charge en mémoire l'enregistrement lié.

Référence

STOCKER SUR LIEN.

STOCKER SUR LIEN (champ)

Paramètre	Type		Description
champ	Champ	→	Champ N

Description

STOCKER SUR LIEN sauvegarde l'enregistrement lié à champ. Vous pouvez exécuter une commande STOCKER SUR LIEN pour mettre à jour un enregistrement créé par CREER SUR LIEN, ou bien lorsque vous voulez sauvegarder des modifications apportées à un enregistrement chargé par CHARGER SUR LIEN.

STOCKER SUR LIEN ne s'applique pas aux sous-tables car la sauvegarde d'un enregistrement parent entraîne automatiquement la sauvegarde des sous-enregistrements.

STOCKER SUR LIEN ne sauvegardera pas un enregistrement verrouillé. Lorsque vous appelez cette commande, vous devez tout d'abord vous assurer que l'enregistrement n'est pas verrouillé. S'il est verrouillé, la commande est ignorée, l'enregistrement n'est pas sauvegardé et aucune erreur ne vous est retournée.

Référence

CHARGER SUR LIEN, CREER SUR LIEN, Enregistrement verrouille, Présentation des triggers.

CHARGER ANCIEN (champ)

Paramètre	Type		Description
champ	Champ	→	Champ N

Description

CHARGER ANCIEN fonctionne de la même manière que CHARGER SUR LIEN, à la différence près que CHARGER ANCIEN utilise la valeur précédente de champ pour établir la relation.

Note : CHARGER ANCIEN utilise l'ancienne valeur du champ N telle qu'elle est retournée par la fonction Ancien. Reportez-vous à la description de cette fonction pour plus d'informations.

CHARGER ANCIEN charge l'enregistrement précédemment lié à l'enregistrement courant. Les champs de cet enregistrement sont alors saisissables. Si vous voulez modifier cet ancien enregistrement lié et le sauvegarder, vous devez appeler la commande STOCKER ANCIEN. Notez que pour un enregistrement venant d'être créé, il n'y pas d'ancien enregistrement lié.

Référence

Ancien, ANCIEN LIEN RETOUR, CHARGER SUR LIEN, STOCKER ANCIEN.

STOCKER ANCIEN (champ)

Paramètre	Type		Description
champ	Champ	→	Champ N

Description

STOCKER ANCIEN fonctionne de la même manière que STOCKER SUR LIEN, à la différence près que STOCKER ANCIEN utilise l'ancien lien vers le champ pour sauvegarder l'ancien enregistrement lié. Avant d'utiliser STOCKER ANCIEN, vous devez charger l'enregistrement avec CHARGER ANCIEN. Utilisez STOCKER ANCIEN lorsque vous voulez sauvegarder des modifications apportées à un enregistrement chargé avec CHARGER ANCIEN.

STOCKER ANCIEN ne sauvegardera pas un enregistrement verrouillé. Lorsque vous appelez cette commande, assurez-vous tout d'abord que l'enregistrement n'est pas verrouillé. S'il est verrouillé, la commande est ignorée, l'enregistrement n'est pas sauvegardé et aucune erreur ne vous est retournée.

Référence

CHARGER ANCIEN, Enregistrement verrouille, Présentation des triggers.

ANCIEN LIEN RETOUR (champ)

Paramètre	Type		Description
champ	Champ	→	Champ recevant un lien

Description

ANCIEN LIEN RETOUR fonctionne comme la commande LIEN RETOUR, à la différence près que ANCIEN LIEN RETOUR utilise l'ancienne valeur du champ pour établir le lien.

Note : ANCIEN LIEN RETOUR utilise l'ancienne valeur du champ N, telle qu'elle est retournée par la fonction Ancien. Reportez-vous à la description de cette fonction pour plus d'informations.

ANCIEN LIEN RETOUR modifie la sélection de la table liée. La commande sélectionne le premier enregistrement de la sélection courante et en fait l'enregistrement courant.

Référence

CHARGER ANCIEN, LIEN RETOUR.

JOINTURE (tableN; table1)

Paramètre	Type		Description
tableN	Table	→	Nom de la table N (d'où part le lien)
table1	Table	→	Nom de la table 1 (où arrive le lien)

Description

La commande JOINTURE crée une nouvelle sélection d'enregistrements dans table1 à partir de la sélection d'enregistrements de la tableN qui lui est liée.

Cette commande ne peut être utilisée que s'il existe un lien de N vers 1. JOINTURE peut opérer au travers de plusieurs niveaux de liens. Il peut y avoir plusieurs tables liées entre la table N et la table 1. Les liens peuvent être manuels ou automatiques.

Attention : N'utilisez pas cette commande dans une transaction.

Exemples

Nous souhaitons trouver tous les clients dont les factures arrivent à échéance aujourd'hui.

(1) L'exemple suivant propose une méthode pour créer une sélection dans la table [Clients] à partir d'une sélection d'enregistrements de la table [Factures] :

```
ENSEMBLE VIDE([Clients];"Paiement Du")
CHERCHER([Factures]; [Factures]PaiementDu=Date du jour)
Tant que (Non(Fin de selection([Factures])))
   CHARGER SUR LIEN([Factures]ClientID)
   AJOINDRE ELEMENT([Clients];"Paiement Du")
ENREGISTREMENT SUIVANT([Factures])
Fin tant que
```

(2) L'exemple suivant parvient au même résultat que le précédent :

```
   CHERCHER([Factures];[Factures]PaiementDu = Date du jour)
⇒  JOINTURE([Factures];[Clients])
```

Référence

CHARGER SUR LIEN, CHERCHER, Présentation des ensembles, SELECTION RETOUR.

SELECTION RETOUR (champ)

Paramètre	Type		Description
champ	Champ	→	Champ de la table N (d'où part le lien)

Description

La commande SELECTION RETOUR crée une sélection d'enregistrements dans la table N, basée sur la sélection courante de la table 1.

Note : SELECTION RETOUR modifie l'enregistrement courant de la table 1.

Attention : N'utilisez pas cette commande dans une transaction.

Exemple

Prenons l'exemple d'une base de données comportant une table [Factures] dont le champ [Factures]IDClient est lié au champ [Clients]NoID de la table [Clients]. L'exemple suivant sélectionne toutes les factures adressées aux clients dont le crédit est supérieur ou égal à 5710 Frs :

```
   ` Sélectionner les clients
CHERCHER ([Clients];[Clients]Credit>=5710)
   `Trouver toutes les factures liées à chacun de ces clients
SELECTION RETOUR ([Factures]NoID)
```

Référence

CHARGER SUR LIEN, CHERCHER, JOINTURE.

32

Listes hiérarchiques

Charger liste (nomListe) → RéfList

Paramètre	Type		Description
nomListe	Alpha	→	Nom de liste créée dans l'éditeur d'énumérations
Résultat	RéfListe	←	Numéro de référence de la liste nouvellement créée

Description

La commande Charger liste crée une liste hiérarchique dont le contenu est copié depuis la liste nomListe créée en mode Structure, dans l'éditeur d'énumérations. La fonction retourne le numéro de référence de la liste nouvellement créée.

Pour savoir si la liste a correctement été chargée, utilisez la fonction Liste existante avec le numéro de référence retourné par LIRE VARIABLES.

Notez que la nouvelle liste est une copie de la liste définie en mode Structure. Par conséquent, toute modification apportée à cette nouvelle liste n'affectera pas la liste définie en mode Structure. De même, toute modification ultérieure de l'énumération n'affecte pas la liste que vous venez de créer.

Si vous modifiez la liste nouvellement créée et voulez enregistrer ces modifications, utilisez la commande STOCKER LISTE.

Si vous n'avez plus besoin de la liste, n'oubliez pas d'appeler SUPPRIMER LISTE pour la supprimer. Sinon, elle reste en mémoire jusqu'à la fin de la session de travail ou jusqu'à ce que le process dans lequel la liste a été créée soit détruit.

Astuce : Si vous associez une liste à un objet de formulaire (liste hiérarchique, onglet ou menu hiérarchique) à l'aide du menu **Enumération** dans la fenêtre des propriétés d'objet, il est inutile d'appeler Charger liste ou SUPPRIMER LISTE dans la méthode de l'objet. 4e Dimension charge et efface la liste automatiquement pour vous.

Exemple

Imaginons que vous créez une base pour le marché international. Vous voulez pouvoir changer la langue utilisée. Dans un formulaire, vous présentez une liste hiérarchique listeHL qui propose les langues disponibles. En mode Structure, vous avez préparé des listes différentes, par exemple “Options US” pour la version anglaise, “Options FR” pour la version française, “Options E/S” pour la version espagnole, etc. De plus, vous maintenez la variable interprocess <>gaLangueCourante dans laquelle vous stockez un code de langue sur 2 caractères, par exemple “US” pour la version anglaise, “FR” pour la version française, “SP” pour la version espagnole, etc. Pour vous assurer que la liste correcte sera chargée en utilisant la langue choisie, vous pouvez écrire :

```
      ` Méthode objet de la liste hiérarchique listeHL
   Au cas ou
      : (Evenement formulaire = Sur chargement)
         C_ENTIER LONG (listeHL)
⇒        listeHL:=Charger liste("Options"+<>gaLangueCourante)
      : (Evenement formulaire = Sur libération)
         SUPPRIMER LISTE(listeHL;*)
   Fin de cas
```

Référence

STOCKER LISTE, SUPPRIMER LISTE.

STOCKER LISTE (liste; nomListe)

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
nomListe	Alpha	→	Nom de la liste tel qu'il doit apparaître dans l'éditeur d'énumérations en mode Structure

Description

La commande STOCKER LISTE sauvegarde la liste dont vous avez passé le numéro de référence dans liste, sous le nom que vous avez passé dans nomListe. La liste est stockée en tant qu'énumération dans l'éditeur d'énumérations du mode Structure.

Si une énumération de même nom existe déjà, son contenu est remplacé.

Référence

Charger liste.

Nouvelle liste → RéfListe

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	RefList	←	Numéro de référence de liste

Description

La commande Nouvelle liste crée une nouvelle liste hiérarchique vide en mémoire et retourne son numéro de référence unique.

ATTENTION : Les listes hiérarchiques résident en mémoire. Une fois que vous en avez terminé avec une liste hiérarchique, il est important que vous l'effaciez à l'aide de la commande SUPPRIMER LISTE. Ainsi, vous libérez la mémoire occupée par la liste hiérarchique dont vous n'avez plus besoin.

D'autres commandes vous permettent de créer des listes hiérarchiques :
• Copier liste crée une nouvelle liste en dupliquant une liste existante.
• Charger liste crée une nouvelle liste en chargeant une énumération créée (manuellement ou par programmation) dans l'éditeur d'énumérations du mode Structure.
• BLOB vers liste crée une nouvelle liste à partir du contenu d'un BLOB dans lequel une liste avait été préalablement stockée.

Une fois que vous avez créé une liste hiérarchique à l'aide de la commande Nouvelle liste, vous pouvez :
• Ajouter des éléments à la liste à l'aide des commandes AJOUTER A LISTE et INSERER ELEMENT.
• Supprimer des éléments de cette liste à l'aide de la commande SUPPRIMER ELEMENT.

Exemple

Reportez-vous à l'exemple de la commande AJOUTER A LISTE.

Référence

AJOUTER A LISTE, BLOB vers liste, Charger liste, Copier liste, INSERER ELEMENT, SUPPRIMER ELEMENT, SUPPRIMER LISTE.

Copier liste (liste) → RéfListe

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de la liste à copier
Résultat	RéfListe	←	Numéro de référence de la nouvelle liste

Description

La commande Copier liste duplique la liste dont vous passez le numéro de référence dans le paramètre liste et retourne le numéro de référence de la nouvelle liste.

Le contenu de la liste copiée est entièrement dupliqué. Une fois que vous en avez terminé avec la copie de la liste, appelez la commande SUPPRIMER LISTE pour l'effacer.

Référence

Charger liste, Nouvelle liste, SUPPRIMER LISTE.

SUPPRIMER LISTE (liste{; *})

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
*		→	Si spécifié, effacer les sous-listes de la mémoire (s'il existe des sous-listes) Si omis, ne pas effacer les sous-listes

Description

La commande SUPPRIMER LISTE efface de la mémoire la liste hiérarchique dont vous avez passé le numéro de référence dans le paramètre liste.

Généralement, vous devez passer le paramètre optionnel *, afin que les sous-listes et les sous-éléments (s'il y en a) rattachés à la liste soient également effacés.

Il n'est pas nécessaire de supprimer une liste associée à un objet de formulaire via la fenêtre de Propriétés des objets : 4D charge et efface la liste automatiquement. Sinon, à chaque fois que vous chargez, copiez, extrayez d'un BLOB ou créez une liste par programmation, appelez la commande SUPPRIMER LISTE lorsque vous n'en avez plus besoin.

Si vous voulez supprimer une sous-liste rattachée à un élément (à tout niveau) d'une autre liste affichée dans un formulaire, procédez de la manière suivante :
1. Appelez INFORMATION ELEMENT avec l'élément parent pour obtenir le numéro de référence de la sous-liste.
2. Appelez CHANGER ELEMENT avec l'élément parent pour dissocier la sous-liste de l'élément de liste avant de l'effacer.
3. Appelez SUPPRIMER LISTE pour effacer la sous-liste dont vous avez obtenu le numéro de référence à l'aide de INFORMATION ELEMENT.
4. Appelez REDESSINER LISTE pour la liste affichée dans le formulaire, de manière à ce que ses éléments et ses sous-listes soient recalculés.

Exemples

(1) Vous disposez, dans votre application, d'une routine de "nettoyage" chargée d'effacer tous les objets et données dont vous n'avez plus besoin lorsque, par exemple, une fenêtre ou un formulaire est refermé(e). A un endroit de cette routine, vous supprimez une liste hiérarchique qui peut avoir déjà été supprimée, suivant les actions de l'utilisateur dans le formulaire. Vous utilisez la fonction Liste existante pour effacer la liste uniquement si c'est nécessaire :

```
      ` Extrait de la sous-routine de nettoyage
   Si (Liste existante(hlList))
⇒     SUPPRIMER LISTE(hlList;*)
   Fin de si
```

(2) Reportez-vous à l'exemple de la fonction Charger liste.

(3) Reportez-vous à l'exemple de la fonction BLOB vers liste.

Référence

BLOB vers liste, Charger liste, Nouvelle liste.

Nombre elements (liste) → Entier long

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
Résultat	Entier long	←	Nombre d'éléments dans la ou les liste(s) déployée(s)

Description

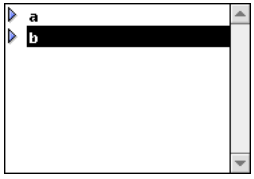
La fonction Nombre elements retourne le nombre d'éléments actuellement “visibles” dans la liste dont vous avez passé le numéro de référence dans liste.

Nombre elements ne retourne pas le nombre total d'éléments de la liste. Cette fonction retourne le nombre d'éléments qui sont visibles, en fonction de l'état déployé/contracté actuel de la liste et de ses sous-listes.

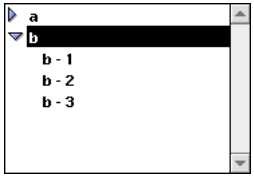
Cette fonction doit être appliquée à une liste affichée dans un formulaire.

Exemples

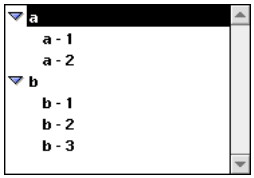
Voici la liste hList affichée en mode Utilisation :



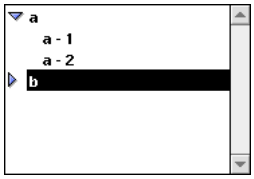
⇒ $vlNbItems:=**Nombre elements**(hList) ` à ce stade, $vlNbItems vaut 2



⇒ $vlNbItems:=**Nombre elements**(hList) ` à ce stade, $vlNbItems vaut 5



⇒ $vlNbItems:=**Nombre elements**(hList) ` à ce stade, $vlNbItems vaut 7



⇒ $vlNbItems:=**Nombre elements**(hList) ` à ce stade, $vlNbItems vaut 4

Référence

Element selectionne, Position element liste.

Liste existante (liste) → Booléen

Paramètre	Type		Description
liste	RéfListe	→	Référence de la liste à tester
Résultat	Booléen	←	Vrai si liste est une liste hiérarchique Faux si liste n'est pas une liste hiérarchique

Description

La fonction Liste existante retourne VRAI si la valeur passée dans le paramètre liste est une référence valide à une liste hiérarchique. Dans les autres cas, elle retourne FAUX.

Exemples

(1) Reportez-vous à l'exemple de la commande SUPPRIMER LISTE.
(2) Reportez-vous aux exemples de la commande PROPRIETES GLISSER DEPOSER.

Référence

PROPRIETES GLISSER DEPOSER.

REDESSINER LISTE (liste)

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste

Description

La commande REDESSINER LISTE recalcule les positions de tous les éléments et sous-listes (s'il y en a) de la liste dont vous avez passé le numéro de référence dans le paramètre liste.

Vous DEVEZ appeler cette commande au moins une fois pour une liste affichée dans un formulaire lorsque vous modifiez un ou plusieurs éléments de la liste ou d'une de ses sous-listes.

Attention

Vous devez passer la véritable instance de la variable liste et non une expression ou autre variable. Si par exemple vous disposez d'une liste hList dans un formulaire :

```
   ` Recalculer la liste après que des modifications aient eu lieu
REDESSINER LISTE (hList) ` BON
   ` ...
$vlList:=hList
   ` ...
   ` Recalculer la liste après que des modifications aient eu lieu
REDESSINER LISTE ($vlList) ` MAUVAIS
   ` ...
```

CHANGER PROPRIETES LISTE (liste; apparence{; icône{; hauteurLigne{; doubleClic}}})

Paramètre	Type		Description
liste	RefListe	→	Numéro de référence de la liste
apparence	Numérique	→	Style graphique de la liste 1 = Liste hiérarchique à la Macintosh 2 = Liste hiérarchique à la Windows
icône	Numérique	→	Référence de ressource MacOS 'cicn' ou 0 = icône par défaut de la plate-forme
hauteurLigne	Numérique	→	Hauteur minimale de la ligne (pixels)
doubleClic	Entier long	→	Déploiement/contraction sur double-clic 0 = autoriser, 1= empêcher

Description

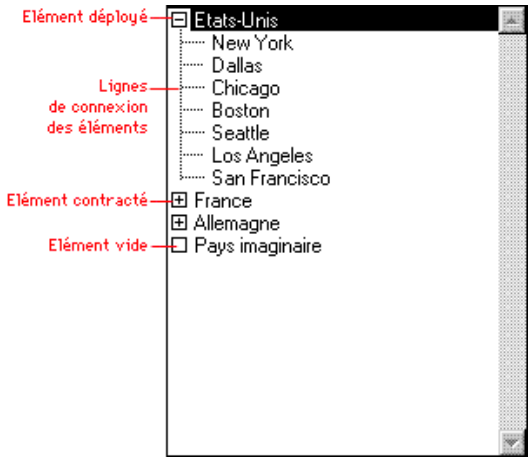
La commande CHANGER PROPRIETES LISTE définit l'apparence et le fonctionnement de la liste hiérarchique dont la référence est passée dans le paramètre liste.

Vous pouvez passer dans le paramètre apparence une des constantes prédéfinies suivantes, fournies par 4e Dimension :

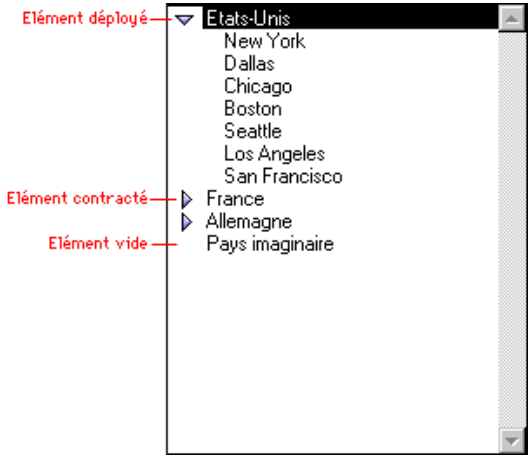
Constante	Type	Valeur
A la Macintosh	Entier long	1
A la Windows	Entier long	2

Avec l'apparence Windows, la liste contient des lignes pointillées qui connectent les éléments et les sous-élements. Une icône "+" signale un élement dont la sous-liste est contractée, une icône "–" signale un élément dont la sous-liste est déployée, une icône vide signale un élément sans sous-élément.

Voici la liste hiérarchique par défaut à la Windows :



Avec l'apparence Macintosh, la liste s'affiche sans lignes pointillées. Une icône en forme de triangle pointant vers la droite signale un élément dont la sous-liste est contractée, une icône en forme de triangle pointant vers le bas signale un élément dont la sous-liste est déployée. Les éléments sans sous-éléments n'ont pas d'icône. Voici la liste hiérarchique par défaut à la Macintosh :



Note : Si vous affichez une liste hiérarchique sans appeler CHANGER PROPRIETES LISTE, la liste s'affiche avec l'apparence par défaut, en fonction la plate-forme sélectionnée en mode Structure dans l'éditeur de formulaires.

Le paramètre icône indique l'icône affichée pour chaque élément. La valeur passée dans icône définit l'icône pour les sous-listes contractées, la valeur icône+1 définit l'icône pour les sous-listes déployées et la valeur icône+2 définit l'icône pour les éléments sans sous-éléments (si l'apparence est à la Windows).

Si, par exemple, vous passez 15000 dans icône, l'icône couleur 'cicn' ID=15000 sera affichée pour sous-liste contractée, l'icône couleur 'cicn' ID=15001 sera affichée pour chaque sous-liste et l'icône couleur 'cicn' ID=15002 sera affichée pour chaque élément ne comportant pas de sous-éléments.

Dans ce cas, il est important de disposer effectivement de ces deux ou trois ressources d'icône couleur 'cicn' dans le fichier de structure de votre base. Si aucune icône couleur n'est présente, les éléments correspondants sont affichés sans icône (à noter que c'est un moyen d'afficher une liste sans icônes).

ATTENTION : Lorsque vous créez des ressources d'icône couleur 'cicn', utilisez des numéros de référence (ID) de ressource supérieurs ou égaux à 15000. Les numéros de référence de ressource inférieurs à 15000 sont réservés à 4e Dimension.

Les numéros de référence par défaut des ressources d'éléments de liste sous MacOS et Windows sont exprimés par les constantes prédéfinies de 4e Dimension suivantes :

Constante	Type	Valeur
Réf icône Macintosh	Entier long	860
Réf icône Windows	Entier long	138

4e Dimension fournit les ressources 'cicn' suivantes :

Numéro d'ID	Description
860	Sous-liste contractée à la Macintosh
861	Sous-liste déployée à la Macintosh
138	Sous-liste contractée à la Windows
139	Sous-liste déployée à la Windows
140	Elément sans sous-liste à la Windows

Si vous n'utilisez pas le paramètre icône, les éléments sont affichés avec les icônes par défaut pour l'apparence choisie.

Les ressources d'icône couleur peuvent avoir des tailles différentes. Vous pouvez créer, par exemple, des icônes couleur 16x16 ou 32x32.
Si vous ne passez pas le paramètre hauteurLigne, la hauteur de ligne d'une liste hiérarchique sera déterminée par la police et la taille de police utilisées pour l'objet. Si vous utilisez des icônes couleurs qui sont trop grandes ou trop larges, elles seront déformées et/ou tronquées par les lignes pointillées (si l'apparence est Windows) ainsi que par le texte des éléments environnants.

Ajustez en conséquence les tailles des icônes couleurs, les polices et les tailles de police. Vous pouvez également passer dans le paramètre hauteurLigne la hauteur minimale des lignes de la liste hiérarchique. Si la valeur que vous passez est supérieure à la hauteur des lignes définie par la police et la taille de police, elle sera utilisée pour fixer la hauteur des lignes.

Note : CHANGER PROPRIETES LISTE affecte l'apparence de la liste, c'est-à-dire les symboles de déploiement/contraction des éléments et les liens entre les éléments. Si vous voulez personnaliser l'icône de chaque élément d'une liste hiérarchique, utilisez la commande CHANGER PROPRIETES ELEMENT.

Le paramètre optionnel doubleClic permet d'empêcher que le double-clic sur un élément de la liste ne provoque le déploiement ou la contraction de sa sous-liste.
Par défaut, une sous-liste est déployée ou contractée en cas de double-clic sur l'élément parent. Certains types d'interfaces peuvent toutefois nécessiter une désactivation de ce fonctionnement. Pour cela, passez 1 dans le paramètre doubleClic. A noter que seul le double-clic sera désactivé. Les sous-listes pourront toujours être déployées ou contractées par un clic sur l'icône de déploiement.
Si vous passez 0 ou omettez ce paramètre, le fonctionnement par défaut est appliqué.

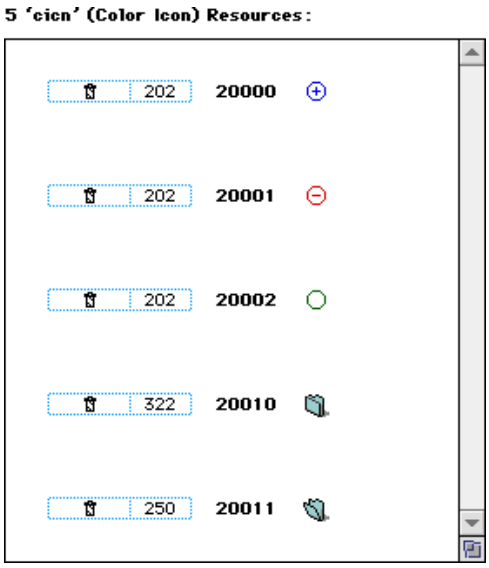
Exemples

La liste hiérarchique suivante a été définie dans l'éditeur d'énumérations en mode Structure :

Dans un formulaire, l'objet liste hiérarchique hlVilles réutilise cette liste avec cette méthode objet :

```
Au cas ou
   : (Evenement formulaire=Sur chargement)
      hlVilles:=Charger liste("Villes")
⇒     CHANGER PROPRIETES LISTE(hlVilles;vlApparence;vlIcon)
   : (Evenement formulaire=Sur libération)
      SUPPRIMER LISTE(hlVilles;*)
Fin de cas
```

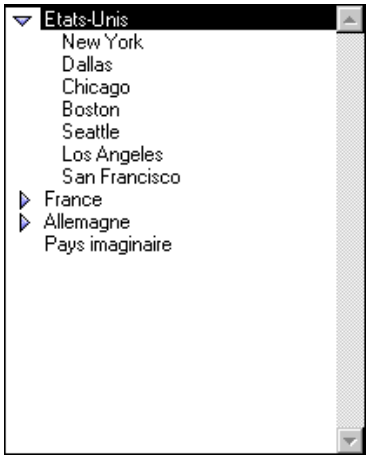
De plus, le fichier de structure de la base a été modifié afin de contenir les ressources d'icônes 'cicn' suivantes :



(1) Avec la ligne suivante :

⇒ **CHANGER PROPRIETES LISTE**(hlVilles;A la Macintosh;Réf icône Macintosh)

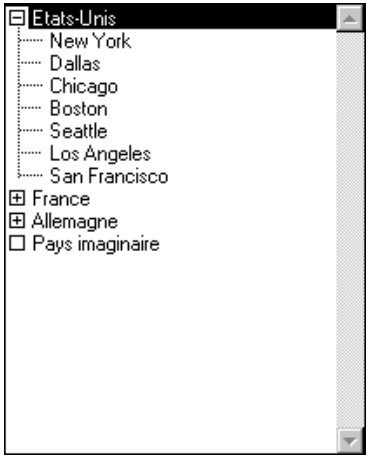
La liste hiérarchique s'affiche ainsi :



(2) Avec la ligne suivante :

⇒ **CHANGER PROPRIETES LISTE**(hlVilles;A la Windows;Réf icône Windows)

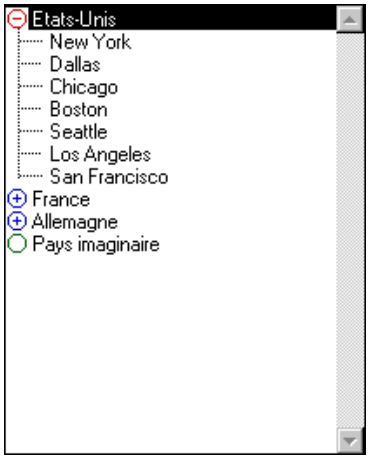
La liste hiérarchique s'affiche ainsi :



(3) Avec la ligne suivante :

⇒ **CHANGER PROPRIETES LISTE**(hlVilles;A la Windows;20000)

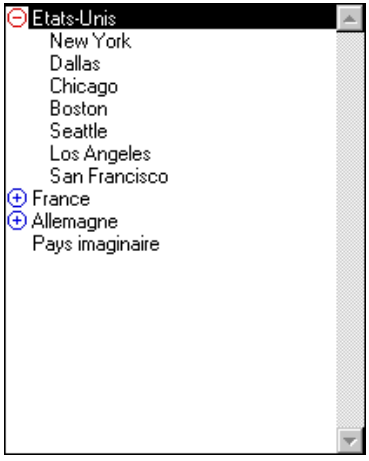
La liste hiérarchique s'affiche ainsi :



(4) Avec la ligne suivante :

⇒ **CHANGER PROPRIETES LISTE**(hlVilles;A la Macintosh;20000)

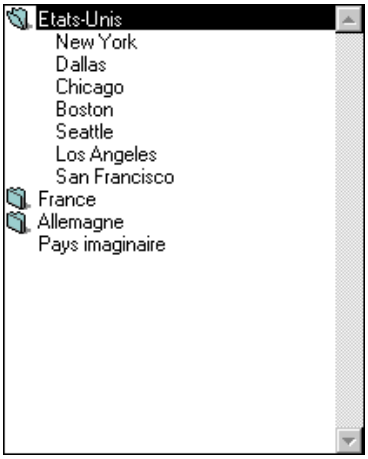
La liste hiérarchique s'affiche ainsi :



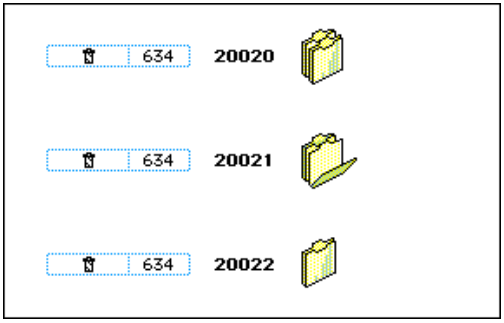
(5) Avec la ligne suivante :

⇒ **CHANGER PROPRIETES LISTE**(hlVilles;A la Macintosh;20010)

La liste hiérarchique s'affiche ainsi :



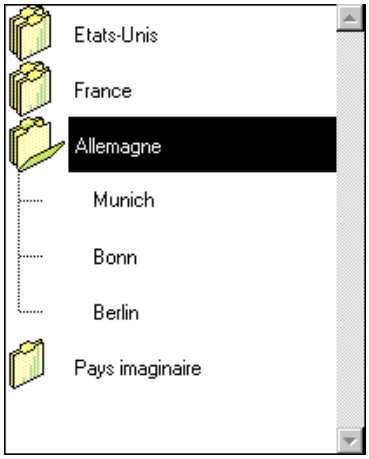
Les ressources d'icône couleur 'cicn' présentées ci-dessous sont ensuite ajoutées au fichier de structure de la base :



(6) Avec la ligne suivante :

⇒ **CHANGER PROPRIETES LISTE**(hlVilles;A la Windows;20020;32)

La liste hiérarchique s'affiche ainsi :



Référence

CHANGER PROPRIETES ELEMENT, LIRE PROPRIETES ELEMENT, LIRE PROPRIETES LISTE.

LIRE PROPRIETES LISTE (liste; apparence{; icône{; hauteurLigne{; doubleClic}}})

Paramètre	Type		Description
liste	RefListe	→	Numéro de référence de la liste
apparence	Numérique	←	Style graphique de la liste 1 = Liste hiérarchique à la Macintosh 2 = Liste hiérarchique à la Windows
icône	Numérique	←	Référence de ressource MacOS 'cicn'
hauteurLigne	Numérique	←	Hauteur minimale de la ligne (pixels)
doubleClic	Entier long	←	Déploiement/contraction sur double-clic 0 = autorisé, 1= empêché

Description

La commande LIRE PROPRIETES LISTE retourne des informations sur la liste hiérarchique dont vous avez passé le numéro de référence dans le paramètre liste.

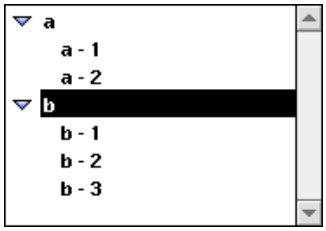
Le paramètre apparence retourne le style graphique de la liste.
Le paramètre icône retourne les icônes utilisées pour symboliser l'état déployé/contracté d'une sous-liste.
Le paramètre hauteurLigne retourne la hauteur de ligne minimale.
Si doubleClic vaut 1, le déploiement ou la contraction des sous-listes en cas de double-clic sur l'élément parent est désactivé. Si doubleClic vaut 0, ce fonctionnement est actif (valeur par défaut).

Ces propriétés peuvent être définies à l'aide de la commande CHANGER PROPRIETES LISTE et/ou dans l'éditeur d'énumérations en mode Structure, si la liste a été créée dans cet éditeur ou sauvegardée avec la commande STOCKER LISTE.

Pour une description complète de ces propriétés d'apparence et de comportement, reportez-vous à la commande CHANGER PROPRIETES LISTE.

Exemple

Voici une liste nommée hList, telle qu'elle apparaît en mode Utilisation :



Voici la méthode objet d'un bouton :

```
      ` Méthode objet du bouton bMacOuWin
⇒     LIRE PROPRIETES LISTE(hList;$vlApparence;$vlIcon;$vlLH)
      Si ($vlApparence=A la Macintosh)
         $vlApparence:=A la Windows
         $vlIcon:=Réf icône Windows
         $vlLH:=20
      Sinon
         $vlApparence:=A la Macintosh
         $vlIcon:=Réf icône Macintosh
         $vlLH:=0
      Fin de si
      CHANGER PROPRIETES LISTE(hList;$vlApparence;$vlIcon;$vlLH)
      REDESSINER LISTE(hList)
```

Cette méthode permet d'afficher la liste ainsi :

Référence

CHANGER PROPRIETES LISTE.

TRIER LISTE (liste{; > ou <})

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
> ou <		→	Ordre de tri : > pour trier la liste dans l'ordre croissant ou < pour trier la liste dans l'ordre décroissant

Description

La commande TRIER LISTE effectue un tri sur la liste dont vous avez passé le numéro de référence dans le paramètre liste.

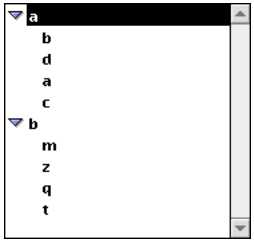
Pour effectuer un tri dans l'ordre croissant, passez > comme deuxième paramètre. Pour effectuer un tri dans l'ordre décroissant, passez < comme deuxième paramètre. Si vous omettez ce paramètre, TRIER LISTE effectue par défaut un tri croissant.

TRIER LISTE trie tous les niveaux de la liste : les éléments de la liste, puis les sous-éléments de chaque sous-liste, puis des sous-listes suivantes, etc., sont triés. C'est pourquoi généralement vous utiliserez la commande TRIER LISTE avec une liste affichée dans un formulaire. Le tri d'une sous-liste a moins d'intérêt car son ordre sera modifié dès qu'un appel à une liste se produira à un niveau supérieur.

TRIER LISTE ne modifie pas l'état courant déployé/contracté de la liste et de ses éventuelles sous-listes, ni l'élément sélectionné. Cependant, comme l'élément sélectionné peut être déplacé à la suite du tri, Element selectionne peut retourner une position différente avant et après le tri.

Exemple

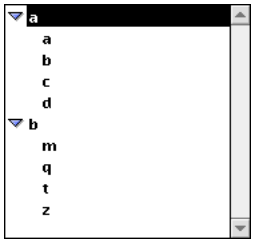
Voici la liste nommée hList, affichée ici en mode Utilisation :



Après l'exécution du code suivant :

```
      ` Trier la liste et ses sous-listes dans l'ordre croissant
⇒     TRIER LISTE(hList;>)
      ` N'oubliez pas d'appeler REDESSINER LISTE sinon elle ne sera pas mise à jour
      REDESSINER LISTE(hList)
```

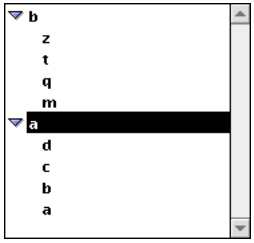
... la liste apparaît ainsi :



Après l'exécution du code suivant :

```
      ` Trier la liste et ses sous-listes dans l'ordre décroissant
⇒     TRIER LISTE(hList;<)
      ` N'oubliez pas d'appeler REDESSINER LISTE sinon elle ne sera pas mise à jour
      REDESSINER LISTE(hList)
```

... la liste apparaît ainsi :



Référence
Element selectionne.

AJOUTER A LISTE (liste; texteElément; numElément{; sous_Liste{; déployée}})

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
texteElément	Alpha	→	Libellé du nouvel élément (255 caractères maximum)
numElément	Numérique	→	Numéro de référence unique du nouvel élément
sous_Liste	RéfListe	→	Sous-liste optionnelle à rattacher au nouvel élément
déployée	Booléen	→	Indique si la sous-liste doit être déployée ou non

Description

La commande AJOUTER A LISTE ajoute un nouvel élément à la liste hiérarchique dont vous avez passé le numéro de référence dans le paramètre liste.

Vous passez le libellé de l'élément dans le paramètre texteElément. Vous pouvez passer une expression de type Alpha ou Texte de 255 caractères maximum. Si vous passez un libellé plus long, il sera tronqué.

Vous passez le numéro de référence unique de l'élément dans le paramètre numElément. Même si nous qualifions ce numéro de référence d'élément comme unique, vous pouvez en réalité passer la valeur que vous voulez. Reportez-vous ci-dessous au paragraphe "Exploiter les numéros de référence des éléments".

Si vous souhaitez également que l'élément comporte des sous-éléments, passez un numéro de référence de liste valide dans le paramètre sous_Liste. Pour que cette sous-liste s'affiche déployée ou contractée, passez respectivement VRAI ou FAUX dans le paramètre déployée.

La référence de la liste que vous passez dans sous_Liste doit être une liste existante. Celle-ci peut être vide, ne contenir qu'un niveau, ou contenir elle-même des sous-listes. Si vous ne voulez pas rattacher de sous-liste au nouvel élément, omettez le paramètre ou passez 0. Si vous passez le paramètre sous_Liste et ne passez pas le paramètre déployée, la sous-liste apparaît par défaut contractée.

Conseils

• Pour insérer un nouvel élément dans une liste, utilisez INSERER ELEMENT. Pour modifier le libellé d'un élément existant ou sa sous-liste, ainsi que son état déployé/contracté, utilisez CHANGER ELEMENT.
• Pour changer l'apparence de l'élément ajouté, utilisez CHANGER PROPRIETES ELEMENT.

Attention : Si vous ajoutez un élément à une liste affichée dans un formulaire ou à une liste rattachée à un élément (à travers un ou plusieurs niveaux) dont la liste est affichée dans un formulaire, vous **devez** appeler REDESSINER LISTE pour que 4D recalcule la liste et l'affiche en fonction de vos modifications. La règle est simple : quel que soit le niveau de la liste sur lequel vous agissez, appliquez REDESSINER LISTE à la liste principale, c'est-à-dire la liste referencée par l'objet dans le formulaire.

Exploiter les numéros de référence des éléments

Chaque élément d'une liste hiérarchique dispose d'un numéro de référence de type Entier long. Cette valeur est destinée uniquement à votre propre usage : 4D ne fait que la maintenir. Voici quelques astuces quant à l'utilisation de ce numéro :

(1) Vous n'avez pas besoin d'identifier chaque élément de façon unique (niveau débutant).

• Premier exemple : vous construisez par programmation un système d'onglets, par exemple, un carnet d'adresses. Comme le système vous retournera le numéro de l'onglet sélectionné, vous n'aurez probablement pas besoin de davantage d'informations. Dans ce cas, ne vous préoccupez pas des numéros de référence des éléments : passez 0 dans le paramètre numElément. Notez que pour un système de carnet d'adresses, vous pouvez prédéfinir une liste A, B,..., Z en mode Structure. Vous pouvez également la créer par programmation afin d'éliminer les lettres pour lesquelles il n'y a pas d'enregistrement.

• Deuxième exemple : en travaillant avec une base, vous construisez progressivement une liste de mots-clés. Vous pouvez sauvegarder la liste à la fin de chaque session, en utilisant les commandes STOCKER LISTE ou LISTE VERS BLOB, et la recharger au début de chaque session, à l'aide des commandes Charger liste ou BLOB vers liste. Vous pouvez afficher cette liste dans une palette flottante ; lorsque l'utilisateur clique sur un mot-clé de la liste, l'élément choisi est inséré dans la zone saisissable sélectionnée du process de premier plan. Vous pouvez également utiliser le glisser-déposer. En tout état de cause, l'important est que vous ne traitez que l'élément sélectionné (par clic ou glisser-déposer), car les commandes Element selectionné (en cas de clic) et PROPRIETES GLISSER DEPOSER vous retournent la position de l'élément que vous devez traiter. En utilisant cette valeur de position, vous obtenez le libellé de l'élément grâce à la commande INFORMATION ELEMENT. Ici aussi, vous n'avez pas besoin d'identifier de façon unique chaque élément ; vous pouvez passer 0 dans le paramètre numElément.

(2) Vous avez besoin d'identifier partiellement les éléments de la liste (niveau intermédiaire).

Vous utilisez le numéro de référence de l'élément pour stocker l'information nécessaire lorsque vous devez agir sur un élément ; ce point est détaillé dans l'exemple fourni plus bas. Dans cet exemple, nous utilisons les numéros de référence des éléments pour stocker des numéros d'enregistrements. Cependant, nous devons pouvoir établir une distinction entre les éléments qui correspondent aux enregistrements [Départements] et ceux qui correspondent aux enregistrements [Employés]. Reportez-vous à l'exemple présenté plus bas.

(3) Vous avez besoin d'identifier les éléments de la liste de façon unique (niveau avancé).

Vous programmez une gestion élaborée de listes hiérarchiques, dans laquelle vous devez absolument pouvoir identifier chaque élément de manière unique à tous les niveaux de la liste. Un moyen simple d'implémenter ce fonctionnement est de maintenir un compteur personnel. Supposons que vous créez une liste hlList à l'aide de la commande Nouvelle liste. A ce stade, vous initialisez un compteur vlhCounter à 0. A chaque fois que vous appelez AJOUTER A LISTE ou INSERER ELEMENT, vous incrémentez ce compteur (vlhCounter:=vlhCounter+1), et vous passez le compteur comme numéro de référence de l'élément. L'astuce consiste à ne pas décrémenter le compteur lorsque vous détruisez des éléments — le compteur ne peut qu'augmenter. En procédant ainsi, vous garantissez l'unicité des numéros de référence des éléments. Puisque ces numéros sont des valeurs de type Entier long, vous pouvez ajouter ou insérer plus de deux milliards d'éléments dans une liste qui a été réinitialisée... (si vous manipulez d'aussi grandes quantités d'éléments, cela signifie généralement que vous devriez utiliser une table plutôt qu'une liste.)

Note : Si vous exploitez les Opérateurs sur les bits, vous pouvez également utiliser les numéros de référence des éléments pour stocker des informations qui peuvent être logées dans un Entier long, c'est-à-dire 2 Entiers, des valeurs de 4 octets ou encore 32 Booléens.

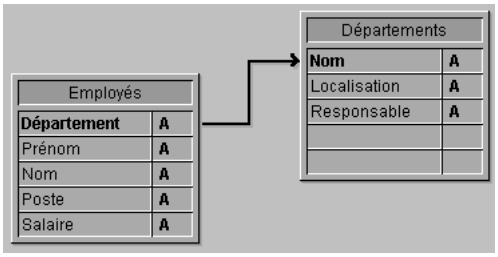
Quand avez-vous besoin de numéros de référence uniques ?

Dans la plupart des cas, lorsque vous utilisez des listes hiérarchiques pour des besoins d'interface utilisateur, pour lesquels seul l'élément sélectionné (par un clic ou par glisser-déposer) est important, vous n'avez pas besoin d'utiliser les numéros de référence des éléments. Les commandes Element selectionne et INFORMATION ELEMENT vous fournissent toutes les informations nécessaires à la gestion de l'élément sélectionné. De plus, des commandes telles que INSERER ELEMENT ou SUPPRIMER ELEMENT vous permettent de manipuler la liste de manière "relative" à l'élément sélectionné.

En pratique, vous devez vous préoccuper des numéros de référence d'éléments lorsque vous voulez accéder directement par programmation à n'importe quel élément de la liste, et pas nécessairement à l'élément couramment sélectionné.

Exemple

Voici une vue partielle de la structure d'une base :



Les tables [Départements] et [Employés] contiennent les enregistrements suivants :

Départements

Nom	Localisation	Responsable
Biologie marine	Vivarium 11	Robert Ibéri
Comptabilité	2ème étage	Louis Berouet
Ventes	RDC ouest	Eliane Bergis

Employés

Département	Prénom	Nom
Biologie marine	Daphné	Vaudelles
Comptabilité	Gérard	Périet
Ventes	Guy	Chartret
Comptabilité	Gilbert	Darieux
Biologie marine	Arnaud	Schmitt
Ventes	Frédérique	Dalhoum
Biologie marine	Pierre	Marty

Vous voulez utiliser une liste hiérarchique, appelée hlList, qui affiche les départements, et pour chaque département, une sous-liste contenant les employés travaillant dans ce département.

La méthode objet de hlList est la suivante:

```
      ` Méthode objet Liste hiérarchique hlList

   Au cas ou

   : (Evenement formulaire=Sur chargement)
      C_ENTIER LONG(hlList;$hSousListe;$vlDépartement;$vlEmployé)
         ` Créer une nouvelle liste hiérarchique vide
      hlList:=Nouvelle liste
         ` Sélectionner tous les enregistrements de la table [Départements]
      TOUT SELECTIONNER([Départements])
         ` Pour chaque Département
      Boucle ($vlDepartement;1;Enregistrements trouves([Départements]))
            ` Sélectionner les employés de ce département
         LIEN RETOUR([Départements]Nom)
            ` Combien sont-ils?
         $vlNbEmployés:=Enregistrements trouves([Employés])
            ` Y-at-il au moins un employé dans ce département?
         Si ($vlNbEmployés>0)
               ` Créer une sous-liste pour l'élément Département
            $hSousListe:=Nouvelle liste
               ` Pour chaque Employé
            Boucle ($vlEmployé;1;Enregistrements trouves([Employés]))
                  ` Ajouter l'élément Employé à la sous-liste
                  ` Noter que le numéro de l'enregistrement [Employés]
                  ` est passé comme numéro de référence de l'élément
⇒               AJOUTER A LISTE($hSousListe;[Employés]NomFamille+", "
                        +[Employés]Prénom;Numero enregistrement([Employés]))
                  ` Aller à l'enregistrement [Employés] suivant
               ENREGISTREMENT SUIVANT([Employés])
            Fin de boucle
         Sinon
               ` Pas d'Employé, pas de sous-liste pour l'élément Département
            $hSousListe:=0
         Fin de si

         ` Ajouter l'élément Département à la liste principale
         ` Notez que le numéro de l'enregistrement [Départements]
         ` est passé comme numéro de référence de l'élément. Le bit #31
         ` du numéro de référence de l'élément est forcé à 1. Ainsi nous pourrons
         ` faire la distinction entre les éléments Département et Employés (cf. note
         ` ci-dessous)
```

```
⇒           AJOUTER A LISTE(hlList;[Départements]Nom;0x80000000 |
                                                    Numero enregistrement([Départements]);
                                                                $hSousListe;$hSousListe#0)
               ` Passer l'élément Département en gras pour renforcer la hiérarchie
               ` de la liste
            CHANGER PROPRIETES ELEMENT(hlList;0;Faux;Gras;0)
               ` Aller au département suivant
            ENREGISTREMENT SUIVANT([Départements])
         Fin de boucle
            ` Trier toute la liste en ordre croissant
         TRIER LISTE(hlList;>)
            ` Afficher la liste en style Windows et forcer la hauteur de ligne minimale
            ` à 14 Pts
         CHANGER PROPRIETES LISTE(hlList;A la Windows;Réf icône Windows;14)

      : (Evenement formulaire=Sur libération)
            ` La liste n'est plus utile. N'oubliez pas de l'effacer !
         SUPPRIMER LISTE(hlList;*)

      : (Evenement formulaire=Sur double clic)
            ` Il y a eu un double-clic
            ` Obtenir la position de l'élément sélectionné
         $vlÉlémentPos:=Element selectionne(hlList)
            ` A toutes fins utiles, vérifier la position
         Si ($vlÉlémentPos # 0)
               ` Obtenir l'information de l'élément de la liste
            INFORMATION ELEMENT(hlList;$vlÉlémentPos;$vlÉlémentRef;
                                                    $vsÉlémentText;$vlÉlémentSousListe;
                                                                $vbÉlémentDéployé)
               ` Cet élément est-il l'élément d'un Département?
            Si ($vlÉlémentRef ?? 31)
                  ` Si oui, c'est un double-clic sur un élément Département
               ALERTE("Vous avez double-cliqué sur l'élément Département "
                                    +Caractere(34)+$vsÉlémentText+Caractere(34)+".")
            Sinon
                  ` Sinon, c'est un double-clic sur un élément Employé. Avec le numéro
                  ` de référence de l'élément parent, trouver l'enregistrement
                  ` [Départements]
               ALLER A ENREGISTREMENT([Départements];Element parent(hlList;
                                                                $vlÉlémentRef)?-31)
```

```
                ` Signaler où l'Employé travaille et de qui il dépend
            ALERTE("Vous avez double-cliqué sur l'élément Employé "+Caractere(34)+
                $vsÉlémentText+Caractere(34)+" qui travaille dans
                le Département "+Caractere(34)+[Départements]Nom+
                Caractere(34)+ " dont le responsable est "+Caractere(34)+
                [Départements]Responsable+Caractere(34)+".")
        Fin de si
    Fin de si
Fin de cas

    ` Note : 4D peut stocker jusqu'à 16 millions d'enregistrements par table
    ` (exactement 16 777 215). Cette valeur est 2^24 moins 1. Le numéro
    ` d'enregistrement tient sur 24 bits. Dans notre exemple, nous utilisons le bit #31
    ` de l'octet supérieur inutilisé pour différencier les éléments des Employés des
    ` Départements.
```

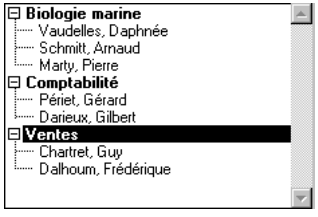
Dans cet exemple, il y a une seule raison d'établir une distinction entre les éléments Départements et les éléments Employés :

1. Nous stockons les numéros d'enregistrements dans les numéros de référence des éléments. En conséquence, nous avons toutes les chances de rencontrer des éléments Départements dont les numéros de référence sont les mêmes que ceux des éléments Employés.

2. Nous utilisons la commande Element parent pour récupérer le parent de l'élément sélectionné. Si nous cliquons sur un élément Employés dont le numéro d'enregistrement associé est 10, et s'il existe aussi un élément Départements qui a le numéro 10, l'élément Départements sera trouvé en premier par Element parent quand cette fonction passera la liste en revue pour repérer l'élément avec le numéro de référence que nous passons. La commande retournera le parent de l'élément Départements et non celui de l'élément Employés.

C'est pourquoi nous avons choisi des numéros de référence d'éléments uniques, non pas pour des questions de principe, mais parce que nous devions différencier les éléments de Départements et d'Employés.

Dans les modes Utilisation et Menus créés, la liste apparaîtra ainsi :



Note : Cet exemple est utile dans le cadre de la gestion de l'interface utilisateur, si vous manipulez un nombre limité d'enregistrements. Souvenez-vous que les listes sont conservées en mémoire ; donc, ne construisez pas d'interfaces utilisateur exploitant des listes hiérarchiques comportant des milliers d'éléments.

Référence

CHANGER ELEMENT, CHANGER PROPRIETES ELEMENT, INSERER ELEMENT.

INSERER ELEMENT (liste; avantElément | *; texteElément; numElément{; sous_Liste{; déployé}})

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
avantElément | *	Num | *	→	Numéro de référence d'élément ou * pour l'élément de la liste actuellement sélectionné
texteElément	Alpha	→	Libellé du nouvel élément (255 caractères maxi)
numElément	Numérique	→	Numéro de référence unique du nouvel élément
sous_Liste	RefList	→	Sous-liste optionnelle rattachée au nouvel élément
déployé	Booléen	→	Indique si la sous-liste doit être déployée ou non

Description

La commande INSERER ELEMENT insère un nouvel élément dans la liste dont le numéro de référence est passé dans liste.

Si vous passez * comme deuxième paramètre, le nouvel élément est inséré avant l'élément actuellement sélectionné dans la liste. Dans ce cas, le nouvel élément devient l'élément actuellement sélectionné.

Sinon, si vous souhaitez insérer le nouvel élément avant un élément spécifique, passez le numéro de référence de cet élément comme deuxième paramètre. Dans ce cas, le nouvel élément inséré n'est pas automatiquement sélectionné. Si le numéro que vous passez ne correspond à aucun élément de la liste, la commande ne fait rien.

Vous passez le texte et le numéro de référence du nouvel élément dans les paramètres texteElément et numElément.

Exemple

L'exemple suivant insère un élément (associé à aucune sous-liste) juste devant l'élément actuellement sélectionné dans la liste hList:

```
   vlUniqueRef:=vlUniqueRef+1
⇒  INSERER ELEMENT(hList;*;"Nouvel élément";vlUniqueRef)
   REDESSINER LISTE(hList)
```

Référence

AJOUTER A LISTE.

CHANGER PROPRIETES ELEMENT (liste; réfElément; saisissable; styles; icône)

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
réfElément	Numérique	→	Numéro de référence d'élément ou 0 pour le dernier élément ajouté à la liste
saisissable	Booléen	→	Vrai = Saisissable, Faux = Non-saisissable
styles	Numérique	→	Style de police pour l'élément
icône	Numérique	→	Numéro de ressource MacOS 'cicn' ou 65536 + numéro de ressource MacOS 'PICT ou 131072 + numéro de référence d'image

Description

La commande CHANGER PROPRIETES ELEMENT modifie l'élément dont vous avez passé le numéro de référence dans réfElément de la liste dont vous avez passé le numéro de référence dans liste.

Si le numéro de référence d'élément ne correspond à aucun élément de la liste, la commande ne fait rien. Vous pouvez passer 0 dans réfElément afin de demander la modification du dernier élément ajouté à la liste (à l'aide de AJOUTER A LISTE).

Si vous travaillez avec les numéros de référence des éléments, assurez-vous d'utiliser des numéros uniques, sinon vous ne pourrez pas différencier les éléments. Pour plus d'informations sur ce point, reportez-vous à la description de la commande AJOUTER A LISTE.

Note : Pour changer le libellé d'un élément ou de ses sous-listes, utilisez la commande CHANGER ELEMENT.

Si vous souhaitez que l'élément soit saisissable, passez Vrai dans le paramètre saisissable, sinon passez Faux.

Important : Pour qu'un élément soit saisissable, il doit appartenir à une liste elle-même saisissable. Pour déclarer une liste saisissable, utilisez la commande CHOIX SAISISSABLE. La commande CHANGER PROPRIETES ELEMENT vous permet de déclarer un élément individuel saisissable ou non. La modification de la propriété saisissable au niveau de la liste ne change pas la propriété saisissable individuelle de chaque élément. Cependant, un élément ne peut être saisissable que si la liste **et** l'élément le sont.

Vous pouvez définir le style de l'élément dans le paramètre styles. Vous passez une ou une combinaison des constantes prédéfinies suivantes :

Constante	Type	Valeur
Standard	Entier long	0
Gras	Entier long	1
Italique	Entier long	2
Souligné	Entier long	4
Contours	Entier long	8
Ombré	Entier long	16
Condensé	Entier long	32
Etendu	Entier long	64

Note : Sous Windows, seuls les styles Standard ou une combinaison de Gras, Italique et Souligné sont disponibles.

Si vous souhaitez associer une icône à l'élément, passez une des valeurs numériques suivantes dans le paramètre icône :
• N, où N est le numéro d'une ressource MacOS 'cicn'
• Utiliser ressource PICT+N, où N est le numéro d'une ressource MacOS 'PICT'.
• Utiliser réf image+N, où N est le numéro de référence d'une image stockée dans la bibliothèque d'images de 4D, en mode Structure.

Si vous ne souhaitez pas associer d'image à l'élément, passez 0 (zéro) dans icône.

Note : Utiliser ressource PICT et Utiliser réf image sont des constantes prédéfinies fournies par 4D.

Exemple

Reportez-vous à l'exemple de la commande AJOUTER A LISTE.

Référence

CHANGER ELEMENT, LIRE PROPRIETES ELEMENT.

LIRE PROPRIETES ELEMENT (liste; réfElément; saisissable{; styles{; icône}})

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
réfElément	Numérique	→	Numéro de référence d'élément
saisissable	Booléen	←	Vrai = Saisissable, Faux = Non-saisissable
styles	Numérique	←	Style de police de l'élément
icône	Numérique	←	Numéro de ressource MacOS 'cicn' ou 65536 + numéro de ressource MacOS 'PICT' ou 131072 + numéro de référence d'image

Description

La commande LIRE PROPRIETES ELEMENT retourne les propriétés de l'élément dont vous avez passé le numéro de référence dans réfElément de la liste dont vous avez passé le numéro de référence dans liste.

Après l'appel :
• saisissable retourne Vrai si l'élément est saisissable.
• styles retourne le style de caractères de l'élément.
• icône retourne l'icône ou l'image associée à l'élément, et 0 s'il n'y en a pas.

Pour plus d'informations sur ces propriétés, reportez-vous à la description de la commande CHANGER PROPRIETES ELEMENT.

Si le numéro de référence d'élément ne correspond à aucun élément de la liste, la commande laisse les paramètres inchangés.

Si vous travaillez avec les numéros de référence des éléments, assurez-vous d'utiliser des numéros uniques, sinon vous ne pourrez pas différencier les éléments. Pour plus d'informations sur ce point, reportez-vous à la description de la commande AJOUTER A LISTE.

Référence

CHANGER ELEMENT, CHANGER PROPRIETES ELEMENT, INFORMATION ELEMENT.

Position element liste (liste; réfElément) → Numérique

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
réfElément	Numérique	→	Numéro de référence d'élément
Résultat	Numérique	←	Position de l'élément parmi la ou les liste(s) déployée(s)

Description

La commande Position element liste retourne la position de l'élément dont vous avez passé le numéro de référence dans réfElément parmi la liste dont vous avez passé le numéro de référence dans liste.

La position est exprimée relativement à l'élément supérieur de la liste, en tenant compte de l'état déployé/contracté de la liste et de ses sous-listes.

Le résultat est donc compris entre 1 et la valeur retournée par Nombre elements.

Si l'élément n'est pas visible car il est inclus dans une liste contractée, Position element liste déploie la liste correspondante de manière à ce que l'élément devienne visible.

Si l'élément n'existe pas, Position element liste retourne 0.

Référence

Nombre elements, SELECTIONNER ELEMENT PAR REFERENCE.

Element parent (liste; réfElément) → Numérique

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
réfElément	Numérique	→	Numéro de référence d'élément
Résultat	Numérique	←	Numéro de référence de l'élément parent ou 0 s'il n'y en a pas

Description

La commande Element parent retourne le numéro de référence de l'élément parent.

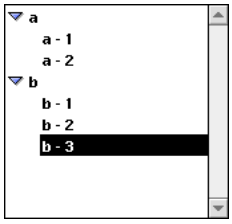
Vous passez un numéro de référence de liste dans liste et le numéro de référence d'un élément de cette liste dans réfElément. En retour, si un élément correspondant à ce numéro de référence existe bien dans la liste et si cet élément se trouve bien dans une sous-liste (et a donc un élément parent), vous récupérez le numéro de référence de l'élément parent.

S'il n'existe pas d'élément de ce numéro ou si cet élément n'a pas d'élément parent, Element parent retourne 0 (zéro).

Si vous travaillez avec les numéros de référence des éléments, assurez-vous d'utiliser des numéros uniques, sinon vous ne pourrez pas différencier les éléments. Pour plus d'informations sur ce point, reportez-vous à la description de la commande AJOUTER A LISTE.

Exemples

Voici la liste hList affichée en mode Utilisation :



Voici de plus les numéros de référence des éléments de cette liste :

Elément	Numéro
a	100
a - 1	101
a - 2	102
b	200
b - 1	201
b - 2	202
b - 3	203

• Avec le code ci-dessous, si l'élément “b - 3” est sélectionné, la variable $vlParentElémRef prend la valeur 200, c'est-à-dire le numéro de référence de l'élément “b” :

```
   $vlElémPos:=Element selectionne(hList)
   INFORMATION ELEMENT(hList;$vlElémPos;$vlElémRef;$vsItemText)
⇒  $vlParentElémRef:=Element parent(hList;$vlElémRef) ` $vlParentElémRef vaut 200
```

• Si l'élément “a - 1” était sélectionné, la variable $vlParentElémRef prendrait la valeur 100, c'est-à-dire le numéro de référence de l'élément “a”.

• Si l'élément “a” ou “b” était sélectionné, la variable $vlParentElémRef prendrait la valeur 0 car ces éléments n'ont pas d'élément parent.

Référence

CHANGER ELEMENT, INFORMATION ELEMENT, Position element liste, SELECTIONNER ELEMENT PAR REFERENCE.

SUPPRIMER ELEMENT (liste; élémentRef | *{; *})

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
élémentRef	*	Num	*	→	Numéro de référence d'élément ou * pour l'élément de la liste actuellement sélectionné
*		→	Si spécifié, effacer les sous-listes de la mémoire (le cas échéant) Si omis, ne pas effacer les sous-listes

Description

La commande SUPPRIMER ELEMENT supprime un élément de la liste dont le numéro de référence est passé dans le paramètre liste.

Si vous passez * comme second paramètre, vous supprimez l'élément actuellement sélectionné de la liste.

Sinon, vous spécifiez le numéro de référence de l'élément à supprimer. Si le numéro ne correspond à aucun élément de la liste, la commande ne fait rien.

Si vous travaillez avec les numéros de référence des éléments, veillez à construire une liste dans laquelle les éléments ont des numéros de référence uniques, sinon vous ne pourrez les différencier. Pour plus d'informations sur ce point, reportez-vous à la description de la commande AJOUTER A LISTE.

Quel que soit l'élément que vous supprimez, vous pouvez passer un troisième paramètre optionnel, *, pour indiquer à 4D de supprimer automatiquement de la mémoire la sous-liste rattachée à l'élément, s'il en existe. Si vous ne passez pas ce paramètre, il est préférable de récupérer au préalable le numéro de référence de la sous-liste (éventuelle) rattachée à l'élément, de manière à pouvoir si besoin est supprimer cette sous-liste à l'aide de la commande SUPPRIMER LISTE.

Exemple

L'exemple suivant supprime l'élément sélectionné de la liste hList. Si une sous-liste est rattachée à l'élément, elle est supprimée (ainsi que toute sous-sous-liste) :

```
⇒ SUPPRIMER ELEMENT(hList;*;*)
     ` N'oubliez pas d'appeler REDESSINER LISTE sinon la liste n'est pas mise à jour
  REDESSINER LISTE(hList)
```

Référence

INFORMATION ELEMENT, SUPPRIMER LISTE.

INFORMATION ELEMENT (liste; positionElém; numElém; textElém{; sous_Liste{; déployé}})

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
positionElém	Numérique	→	Position de l'élément dans la ou les liste(s) déployée(s)
numElém	Numérique	←	Numéro de référence de l'élément
textElém	Alpha	←	Libellé de l'élément
sous_Liste	RéfListe	←	Numéro de référence de sous-liste (s'il y en a)
déployé	Booléen	←	Si une sous-liste est rattachée à l'élément : Vrai = la sous-liste est déployée Faux = la sous-liste est contractée

Description

La commande INFORMATION ELEMENT retourne des informations sur l'élément dont vous avez passé la position dans le paramètre positionElém parmi la liste dont vous avez passé le numéro de référence dans liste.

La position doit être exprimée relativement à l'état déployé/contracté de la liste et de ses sous-listes. Vous devez passer une valeur de position comprise entre 1 et la valeur retournée par Nombre elements. Si vous passez une valeur située hors de cet intervalle, INFORMATION ELEMENT retourne les paramètres inchangés.

Après l'appel, vous récupérez :
• Le numéro de référence de l'élément dans numElém.
• Le libellé de l'élément dans textElém.

Si vous passez les paramètres optionnels sous_Liste et déployé :
• sous_Liste contient le numéro de référence de la sous-liste rattachée à l'élément. Si l'élément n'a pas de sous-liste associée, sous_Liste retourne zéro.
• Si l'élément comporte une sous-liste, déployé retourne Vrai si la sous-liste est déployée, et Faux sinon.

Exemples

(1) En partant de l'hypothèse que hList est une liste dont les éléments ont des numéros de référence uniques, le code suivant inverse automatiquement l'état déployé/contracté de la sous-liste, si elle existe, rattachée à l'élément sélectionné :

```
C_BOOLEEN($vbDéployé)
C_ENTIER LONG($hSousListe;$vlElemRef)
C_ALPHA(31;$vsElemText)
   `La déclaration de ces variables est nécessaire si vous souhaitez compiler la méthode

$vlElemPos:=Element selectionne(hList)
Si ($vlElemPos>0)
⇒     INFORMATION ELEMENT(hList;$vlElemPos;$vlElemRef;$vsElemText;$hSousListe;
                                                                  $vbDéployé)
   Si (Liste existante($hSousListe))
      CHANGER ELEMENT(hList;$vlElemRef;$vsElemText;$hSousListe;
                                                        Non($vbDéployé))
      REDESSINER LISTE(hList)
   Fin de si
Fin de si
```

(2) Reportez-vous à l'exemple de la commande AJOUTER A LISTE.

Référence

CHANGER ELEMENT, CHANGER PROPRIETES ELEMENT, Element parent, Element selectionne, LIRE PROPRIETES ELEMENT, Position element liste.

CHANGER ELEMENT (liste; réfElément; textElément; nouvelRéf{; sous_Liste; déployée})

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
réfElément	Numérique	→	Numéro de référence d'élément ou 0 pour le dernier élément ajouté à la liste
textElément	Alpha	→	Nouveau libellé d'élément
nouvelRéf	Numérique	→	Nouveau numéro de référence d'élément
sous_Liste	RéfListe	→	Nouvelle sous-liste rattachée à l'élément ou 0 = pas de sous-liste (détacher sous-liste courante) ou -1 = pas de changement
déployée	Booléen	→	Indique si la sous-liste doit être déployée/contractée

Description

La commande CHANGER ELEMENT modifie l'élément dont vous avez passé le numéro de référence dans réfElément de la liste dont vous avez passé le numéro de référence dans liste.

Si le numéro de référence d'élément ne correspond à aucun élément de la liste, la commande ne fait rien. Vous pouvez passer 0 dans réfElément afin de demander la modification du dernier élément ajouté à la liste (à l'aide de AJOUTER A LISTE).

Si vous travaillez avec les numéros de référence des éléments, assurez-vous d'utiliser des numéros uniques, sinon vous ne pourrez pas différencier les éléments. Pour plus d'informations sur ce point, reportez-vous à la description de la commande AJOUTER A LISTE.

Vous pouvez passer le nouveau libellé de l'élément dans le paramètre textElément. Si vous souhaitez changer le numéro de référence de l'élément, passez la nouvelle valeur dans le paramètre nouvelRéf, sinon passez la même valeur que dans réfElément.

Si vous voulez associer une sous-liste à l'élément, passez le numéro de référence de la sous-liste dans le paramètre sous_Liste. Dans ce cas, vous devez également spécifier si la nouvelle sous-liste devra apparaître déployée ou contractée en passant respectivement Vrai ou Faux dans le paramètre déployée.

Si vous voulez dissocier de l'élément une sous-liste qui lui est actuellement rattachée, passez 0 (zéro) dans sous_Liste. Dans ce cas, il est conseillé d'avoir préalablement obtenu le numéro de référence de cette liste à l'aide de la commande INFORMATION ELEMENT, afin de pouvoir effacer la sous-liste avec la commande SUPPRIMER LISTE si vous n'en avez plus besoin.

Si vous ne souhaitez pas modifier les propriétés de sous-liste de l'élément, passez -1 dans le paramètre sous_Liste.

Exemples

(1) Nous supposons que hList est une liste dont les éléments ont des numéros de référence uniques. La méthode objet suivante d'un bouton ajoute une sous-liste à l'élément actuellement sélectionné dans la liste hList :

```
$vlItemPos:=Element selectionne(hList)
Si ($vlItemPos>0)
   INFORMATION ELEMENT(hList;$vlItemPos;$vlItemRef;$vsItemText;$hSouslist;
                                                                $vbExpanded)
   $vbNouvSousList:=Non(Liste existante($hSouslist))
   Si ($vbNouvSousList)
      $hSouslist:=Nouvelle liste
   Fin de si
   vlUniqueRef:=vlUniqueRef+1
   AJOUTER A LISTE($hSousList;"Nouvel élément";vlUniqueRef)
   Si ($vbNouvSousList)
      CHANGER ELEMENT(hList;$vlItemRef;$vsItemText;$vlItemRef;$hSouslist;Vrai)
   Fin de si
   SELECTIONNER ELEMENT PAR REFERENCE(hList;vlUniqueRef)
   REDESSINER LISTE(hList)
Fin de si
```

(2) Reportez-vous à l'exemple de la commande INFORMATION ELEMENT.
(3) Reportez-vous à l'exemple de la commande AJOUTER A LISTE.

Référence

CHANGER PROPRIETES ELEMENT, INFORMATION ELEMENT, LIRE PROPRIETES ELEMENT.

Element selectionne (liste) → Entier long

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
Résultat	Entier long	←	Position de l'élément de liste sélectionné parmi la ou les liste(s) déployée(s)

Description

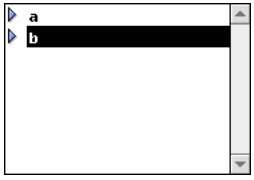
La fonction Element selectionne retourne la **position** de l'élément sélectionné dans la liste dont vous avez passé le numéro de référence dans le paramètre liste.

Cette fonction doit être appliquée à une liste affichée dans un formulaire afin de détecter l'élément sélectionné par l'utilisateur.

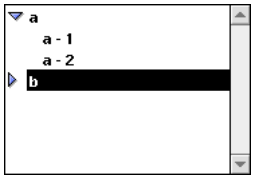
Si la liste comporte des sous-listes, appliquez la fonction à la liste principale (celle qui est associée au formulaire), et non à une de ses sous-listes. La position est exprimée relativement à l'élément supérieur de la liste principale, en tenant compte de l'état déployé/contracté en cours de la liste et de ses sous-listes.

Exemple

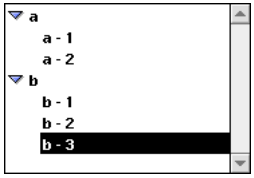
Voici la liste hList telle qu'elle apparaît en mode Utilisation :



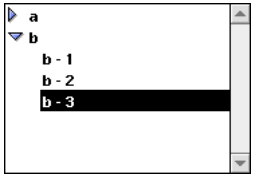
⇒ $vlItemPos:=**Element selectionne**(hList) ` à ce stade, $vlItemPos vaut 2



⇒ $vlItemPos:=**Element selectionne**(hList) ` à ce stade, $vlItemPos vaut 4



⇒ $vlItemPos:=**Element selectionne**(hList) ` à ce stade, $vlItemPos vaut 7



⇒ $vlItemPos:=**Element selectionne**(hList) ` à ce stade, $vlItemPos vaut 5

Référence

SELECTIONNER ELEMENT, SELECTIONNER ELEMENT PAR REFERENCE.

SELECTIONNER ELEMENT (liste; positionElém)

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
positionElém	Numérique	→	Position de l'élément dans la ou les liste(s) déployée(s)

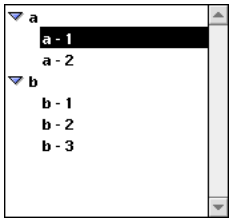
Description

La commande SELECTIONNER ELEMENT sélectionne l'élément dont vous avez passé la position dans positionElém à l'intérieur de la liste dont vous avez passé le numéro de référence dans liste.

Le paramètre positionElém représente la position de l'élément exprimée en tenant compte de l'état déployé/contracté de la liste et de ses sous-listes. Passez une position comprise entre 1 et la valeur retournée par Nombre elements. Si vous passez une valeur située en-dehors de cet intervalle, le premier élément est sélectionné par défaut.

Exemples

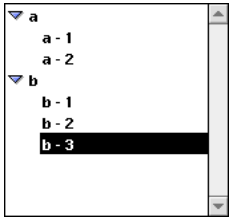
Voici la liste hList affichée en mode Utilisation :



Après l'exécution des lignes de code suivantes :

```
⇒ SELECTIONNER ELEMENT(hList;Nombre elements(hList))
  REDESSINER LISTE(hList)x
```

... le dernier élément visible est sélectionné :



Référence

Element selectionne, SELECTIONNER ELEMENT PAR REFERENCE.

SELECTIONNER ELEMENT PAR REFERENCE (liste; réfElément)

Paramètre	Type		Description
liste	RéfListe	→	Numéro de référence de liste
réfElément	Numérique	→	Numéro de référence d'élément

Description

La commande SELECTIONNER ELEMENT PAR REFERENCE sélectionne l'élément dont vous avez passé le numéro de référence dans réfElément parmi la liste dont vous avez passé le numéro de référence dans liste. Si le numéro d'élément ne correspond à aucun élément de la liste, la commande ne fait rien.

Si l'élément n'est pas visible (car il est par exemple inclus dans une liste contractée), SELECTIONNER ELEMENT PAR REFERENCE déploie la ou les sous-liste(s) correspondante(s) de manière à ce que le nouvel élément sélectionné devienne visible.

Si vous travaillez avec les numéros de référence des éléments, assurez-vous d'utiliser des numéros uniques, sinon vous ne pourrez pas différencier les éléments. Pour plus d'informations, reportez-vous à la description de la commande AJOUTER A LISTE.

Exemple

En supposant que hList est une liste dont les éléments ont des numéros de référence uniques, la méthode objet de bouton suivante sélectionne l'élément parent (s'il existe) de l'élément actuellement sélectionné :

```
    ` Récupérer la position de l'élément sélectionné
$vlElémPos:=Element selectionne(hList)
INFORMATION ELEMENT(hList;$vlElémPos;$vlElémRef;$vsElémText) ` N° de l'élément
    ` Numéro de l'élément parent (s'il existe)
$vlParentElémRef:=Element parent(hList;$vlElémRef)
Si ($vlParentElémRef>0)
        ` Sélection de l'élément parent
⇒   SELECTIONNER ELEMENT PAR REFERENCE(hList;Element parent(hList;
                                                    $vlElémRef))
    REDESSINER LISTE(hList) ` Ne pas oublier de redessiner la liste pour la mettre à jour
Fin de si
```

Référence

Element selectionne, SELECTIONNER ELEMENT.

33

Menus

Terminologie : La documentation sur les commandes de menus emploie indifféremment **commande de menu** et **ligne de menu** lorsqu'elle évoque une ligne d'un menu.

Barres de menus

Les barres de menus sont identifiées par des numéros plutôt que par des noms. La première barre de menus est la barre de menus "Barre n°1". C'est aussi la barre de menus par défaut. Si vous souhaitez ouvrir une application avec une barre de menus autre que la barre n°1, vous devez appeler la commande CHANGER BARRE dans la Méthode base Sur ouverture.

Chaque commande de menu peut être associée à une méthode projet. Si vous n'affectez pas de méthode à une commande de menu, la sélection de cette commande de menu provoque la fermeture du mode Menus créés et le passage en mode Utilisation. Si l'utilisateur exploite une application en menus créés uniquement ou ne dispose pas des privilèges d'accès pour le mode Utilisation, cela provoquera la fermeture de l'application.

Chaque barre de menus comporte par défaut trois menus — **Fichier**, **Edition**, et **Aide** (sous Windows) ou **Pomme**, **Fichier** et **Edition** (sous MacOS).

• Le menu **Fichier** ne contient qu'une commande de menu : **Quitter**. La commande de menu Quitter n'a aucune méthode associée ; c'est de cette manière qu'elle indique à 4e Dimension qu'il faut quitter le mode Menus créés. Vous pouvez renommer le menu Fichier, lui ajouter des commandes de menu ou le garder tel quel. S'il est renommé, il n'apparaîtra plus sur la gauche de la barre de menus. Il est recommandé de toujours garder la commande de menu Quitter comme dernière commande du menu Fichier.

• Le menu **Edition** contient les commandes de menu d'édition standard. Le menu Edition n'est pas affiché dans l'éditeur de menus et ne peut pas être modifié.

• Sous Windows, le menu **Aide** contient la ligne de menu **A propos de 4e Dimension** ainsi que toutes les commandes d'appel des aides en ligne de l'application. Le menu **Aide** n'est pas affiché dans l'éditeur de menus et ne peut pas être modifié, hormis pour la commande d'A propos, qui peut être modifiée à l'aide de la commande APPELER SUR A PROPOS.

• Sous MacOS, le menu **Pomme** contient la commande d'A propos ainsi que toutes les applications qui sont placées dans le dossier Menu Pomme du dossier Système. Le menu Pomme n'est pas affiché dans l'éditeur de menus et ne peut pas être modifié, hormis pour la commande d'A propos, qui peut être modifiée à l'aide de la commande APPELER SUR A PROPOS.

Important : Les barres de menus sont "interprocess". Toute modification effectuée sur une barre sera répercutée dans tous les process où la barre est utilisée.

Numéros des menus et des commandes de menu

Comme les barres de menus, les menus sont numérotés. Les menus **Edition**, **Aide** et **Pomme** ne sont pas pris en compte car ils ne peuvent pas être modifiés. Le menu Fichier est le menu 1. Les autres menus sont numérotés séquentiellement de gauche à droite (2, 3, 4, etc.). La numérotation des menus est importante lorsque vous travaillez, par exemple, avec la fonction Menu choisi.
Lorsqu'un menu est associé à un formulaire, le principe de numérotation est différent. Le premier menu ajouté commence avec le numéro 2049. Pour référencer un menu associé à un formulaire, ajoutez 2048 au numéro initial du menu.

Les commandes de chacun des menus sont numérotées séquentiellement de haut en bas. La commande supérieure a le numéro 1.

Les barres de menus associées

Vous pouvez associer une barre de menus à un formulaire à l'aide de la commande **Associer une barre de menu** du menu **Formulaire** de l'Editeur de formulaires. La barre est alors appelée "barre de menus de formulaire" en ce qui concerne les points suivants.

Les menus d'un formulaire sont ajoutés à la barre de menus courante lorsque le formulaire est affiché. Les menus sont ajoutés pour les formulaires entrée dans les modes Utilisation et Menus créés ainsi que pour les formulaires de sortie dans le mode Menus créés.

Les barres de menus de formulaires sont référencées par un numéro de barre. Si le numéro de la barre de menus affiché avec le formulaire courant est le même que celui de la barre de menus associée au formulaire, cette dernière ne s'affiche pas.

Si vous spécifiez un numéro négatif pour une barre de menus, 4e Dimension utilise la valeur absolue de la barre de menu. Par exemple, si vous spécifiez –3 comme numéro de barre de menus, la barre de menus n°3 est utilisée. Cependant, lorsque la barre de menus d'un formulaire a été spécifiée avec un numéro négatif, les commandes de menus de tous les menus de la barre (écran d'accueil et formulaire) exécuteront les méthodes qui leurs sont associées.

Si vous ne spécifiez pas de numéro négatif pour une barre de menus de formulaire, la sélection d'une de ses commandes de menus n'exécutera pas la méthode qui lui est associée ; au lieu de cela, un événement Sur menu sélectionné sera envoyé à la méthode formulaire, et vous pourrez utiliser la commande Menu choisi pour tester le menu sélectionné.

Modifier des commandes de menu par programmation

4e Dimension fournit les commandes suivantes pour ajouter, supprimer, insérer ou modifier des commandes d'un menu de la barre affichée ou installée dans un process :

- ACTIVER LIGNE MENU
- INACTIVER LIGNE MENU
- CHANGER TEXTE LIGNE MENU
- CHANGER STYLE LIGNE MENU
- MARQUER LIGNE MENU
- CHANGER TEXTE LIGNE MENU
- AJOUTER LIGNE MENU
- INSERER LIGNE MENU
- SUPPRIMER LIGNE MENU

L'aire d'action de ces commandes est la barre de menus courante. Dès que vous appelez une nouvelle fois CHANGER BARRE, tous les menus et les commandes de menus retrouveront leur état originel tel qu'il est défini dans l'éditeur de barres de menus du mode Structure.

Vous devez passer à chacune de ces commandes un numéro de menu et un numéro de commande.

Comme expliqué plus haut, les menus sont numérotés de 1 à N de gauche à droite. Par exemple, le menu Fichier est généralement le premier menu. Sous Windows, les fichiers **Edition** et **Aide** sont exclus de cette numérotation. Sous MacOS, les menus **Pomme** et **Fichier** sont exclus.

Il est à noter que la commande Nombre de menus ne tient pas compte de ces menus. Si, par exemple, votre barre de menus est constituée de trois menus Fichier, Clients et Factures, Nombre de menus retournera 3 (en ignorant les menus système maintenus par 4D).

Les commandes de menus sont numérotées de 1 à N de haut en bas, y compris les séparateurs.

Les menus insérés dans la barre de menus au moyen d'une barre de menus associée à un formulaire (et donc ajoutée à la barre de menus courante) sont numérotés de gauche à droite et commencent par le numéro 2049 (2048 + 1 à N).

La commande Menu choisi retourne les numéros de menus et de commande en respectant cette convention.

Important : Ces commandes n'ont pas accès aux menus système maintenus par 4D (Edition et Aide sous Windows, Pomme et Edition sous MacOS).

Menus connectés : Les menus peuvent être connectés à des barres de menu. Si un menu connecté est modifié à l'aide d'une de ces commandes, chacune des instances de ce menu reflètera ces modifications. Pour plus d'informations sur ce point, reportez-vous au manuel *Mode Structure* de 4e Dimension.

CHANGER BARRE (barreNum{; process}{; *})

Paramètre	Type		Description
barreNum	Numérique	→	Numéro de la barre de menus
process	Numérique	→	Numéro de référence du process
*		→	Conserver l'état de la barre de menus

Description

La commande CHANGER BARRE remplace la barre de menus courante par la barre de menus barreNum, pour le process en cours uniquement.

Si vous passez le paramètre optionnel process, c'est la barre de menus du process spécifié qui sera remplacée par la barre barreNum.
Le paramètre optionnel * vous permet de conserver l'état de la barre de menus. Si ce paramètre est omis, CHANGER BARRE réinitialise la barre de menus lors de l'exécution de la commande.

Imaginez, par exemple, que l'instruction CHANGER BARRE(1) soit exécutée. Ensuite, plusieurs commandes de menu sont désactivées à l'aide de la commande INACTIVER LIGNE. Si CHANGER BARRE(1) est exécutée une seconde fois, soit à partir du même process, soit à partir d'un autre process, toutes les commandes de menu retournent à leur état d'activation initial.
Si CHANGER BARRE(1;*) est exécutée, la barre de menus conservera son état précédent, les commandes de menu qui étaient inactivées le resteront.

Note : Si vous ne passez pas le paramètre optionnel process, '*' peut être le second paramètre. Autrement dit, CHANGER BARRE(1;2;*) et CHANGER BARRE(1;*) sont deux syntaxes valides.

Lorsqu'un utilisateur arrive en mode Menus créés, la première barre de menus s'affiche (Barre de menus n° 1). Vous pouvez changer ce numéro de barre de menus par défaut en spécifiant la barre que vous voulez dans la Méthode base Sur ouverture, ou dans la méthode de démarrage associée à un utilisateur.

Exemples

(1) L'exemple suivant remplace la barre de menus courante par la barre de menus n° 3 et initialise l'état des commandes des menus :

⇒ **CHANGER BARRE** (3)

(2) L'exemple suivant remplace la barre de menus courante par la barre de menus n° 3 et conserve l'état des commandes des menus : celles qui étaient précédemment inactivées apparaîtront inactivées.

⇒ **CHANGER BARRE** (3;*)

(3) L'exemple suivant remplace la barre de menus courante par la barre de menus n° 3 pendant que des enregistrements sont en cours de modification. Une fois les enregistrements modifiés, la barre de menus n° 2 est réaffichée. L'état des commandes de ce menu est conservé :

```
⇒ CHANGER BARRE(3)  ` Définir la barre de menus n° 3 pour le formulaire suivant
   TOUT SELECTIONNER([Clients])
   MODIFIER SELECTION([Clients])  ` Afficher la sélection
⇒ CHANGER BARRE(2;*)  ` Après modification, retour à la barre de menus n° 2
```

Référence

Gestion des menus.

CACHER BARRE DE MENUS

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande CACHER BARRE DE MENUS rend invisible la barre de menus.

Si la barre de menus était déjà cachée, la commande est sans effet.

Exemple

La méthode suivante passe un enregistrement en plein écran (sous MacOS) jusqu'à ce que l'utilisateur clique avec le bouton de la souris :

```
      CACHER BARRE OUTILS
⇒     CACHER BARRE DE MENUS
      Creer fenetre (-1;-1;1 + Largeur ecran; 1 + Hauteur ecran ;
                                                Autre boîte de dialogue modale)
      FORMULAIRE ENTREE([Tableaux];"Plein écran 800")
      AFFICHER ENREGISTREMENT([Tableaux])
      Repeter
         POSITION SOURIS($vlX;$vlY;$vlBouton)
      Jusque($vlBouton#0)
      FERMER FENETRE
      AFFICHER BARRE DE MENUS
      AFFICHER BARRE OUTILS
```

Note : Sous Windows, la taille de la fenêtre sera limitée par celle de la fenêtre de l'application.

Référence

AFFICHER BARRE DE MENUS, AFFICHER BARRE OUTILS, CACHER BARRE OUTILS.

AFFICHER BARRE DE MENUS

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

La commande AFFICHER BARRE DE MENUS rend visible la barre de menus.

Si la barre de menus était déjà visible, cette commande ne fait rien.

Exemple

Reportez-vous à l'exemple de la commande CACHER BARRE DE MENUS.

Référence

AFFICHER BARRE OUTILS, CACHER BARRE DE MENUS, CACHER BARRE OUTILS.

APPELER SUR A PROPOS

version 5

APPELER SUR A PROPOS (libelléLigne; méthode)

Paramètre	Type		Description
libelléLigne	Alpha	→	Nouvelle ligne de menu A propos...
méthode	Alpha	→	Méthode à exécuter lorsque la ligne est choisie

Description

La commande APPELER SUR A PROPOS remplace la ligne de menu **A propos de 4e Dimension** du menu **Aide** (sous Windows) ou du menu **Pomme** (sous MacOS) par libelléLigne.

Après l'appel de cette commande, lorsqu'un utilisateur sélectionne la ligne de menu, la méthode méthode est appelée. Cette méthode peut afficher une boîte de dialogue qui fournit des informations sur les versions de votre base.

L'icône de 4e Dimension, le numéro de version de 4e Dimension, le numéro de version de 4D Compiler et une ligne de copyrights seront affichés en haut de la boîte de dialogue.

Note : Le numéro de version de compilation de votre application est aussi affiché si vous avez activé l'option **Version automatique** dans votre projet 4D Compiler.

Exemples

(1) L'exemple suivant remplace la commande de menu **A propos** par la commande de menu **A propos du programmateur**. La méthode A PROPOS affiche une fenêtre d'A propos personnalisée :

```
⇒ APPELER SUR A PROPOS(“A propos du programmateur...”;“A PROPOS”)
```

(2) L'exemple suivant réinitialise la commande de menu d'A propos de 4e Dimension :

```
⇒ APPELER SUR A PROPOS("A propos de 4e Dimension®";"")
```

Menu choisi → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Commande de menu sélectionnée Mot machine haut = n° de menu Mot machine bas = n° de commande de menu

Description

Menu choisi ne s'utilise que lorsqu'un formulaire est affiché. Cette fonction détecte la commande de menu choisie dans un menu.

Astuce : A chaque fois que cela est possible, utilisez des méthodes associées à des commandes de menus dans une barre associée (avec un numéro de barre négatif) plutôt que d'appeler Menu choisi. Les barres de menus associées sont plus faciles à gérer, puisqu'il n'est pas nécessaire de tester leur sélection. Cependant, si vous utilisez les commandes AJOUTER LIGNE MENU ou INSERER LIGNE MENU, vous devez utiliser Menu choisi car les lignes de menus créées de cette manière ne sont associées à aucune méthode.

Menu choisi retourne le numéro système du menu sélectionné sous forme d'Entier long. Pour obtenir le numéro du menu, divisez Menu choisi par 65536 et convertissez le résultat en Entier. Pour obtenir le numéro de la commande de menu, calculez le modulo de Menu choisi avec le coefficient 65536. Utilisez les formules suivantes pour calculer le numéro du menu et de la commande de menu :

⇒ Menu := **Menu choisi** \ 65536
⇒ Ligne de menu := **Menu choisi** % 65536

A partir de la version 6 de 4D, vous pouvez également extraire ces valeurs à l'aide des Opérateurs sur les bits, comme dans l'exemple suivant :

⇒ Menu := (**Menu choisi** & 0xFFFF0000) >> 16
⇒ Ligne de menu := **Menu choisi** & 0xFFFF

Menu choisi retourne 0 si aucune commande de menu n'est sélectionnée.

Exemple

La méthode formulaire suivante utilise la fonction Menu choisi pour fournir les arguments "menu" et "ligne de menu" à MARQUER LIGNE :

```
      Au cas ou
         : (Evenement formulaire=Sur menu sélectionné)
⇒           Si (Menu choisi # 0)
⇒              MARQUER LIGNE MENU (Menu choisi \ 65536 ; Menu choisi % 65536;
                                                          Caractere (18))
           Fin de si
      Fin de cas
```

Référence

Gestion des menus.

Nombre de menus

Nombre de menus {(process)} → Numérique

Paramètre	Type		Description
process	Numérique	→	Numéro de référence de process
Résultat	Numérique	←	Nombre de menus de la barre de menus courante

Description

La commande Nombre de menus retourne le nombre de menus présents dans la barre de menus.

Si vous omettez le paramètre process, Nombre de menus s'applique à la barre de menus du process courant. Sinon, Nombre de menus s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Référence

Nombre de lignes du menu.

Nombre de lignes du menu

version 6.0

Nombre de lignes du menu (menu{; process}) → Numérique

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
process	Numérique	→	Numéro de référence de process
Résultat	Numérique	←	Nombre de lignes du menu

Description

La commande Nombre de lignes du menu retourne le nombre de lignes (commandes) de menus présentes dans le menu dont vous avez passé le numéro dans menu.

Si vous omettez le paramètre process, Nombre de lignes du menu s'applique à la barre de menus du process courant. Sinon, Nombre de lignes du menu s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Référence

Nombre de menus.

Titre menu (menu{; process}) → Alpha

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
process	Numérique	→	Numéro de référence de process
Résultat	Alpha	←	Titre du menu

Description

La commande Titre menu retourne le titre du menu dont vous avez passé le numéro dans menu.

Si vous omettez le paramètre process, Titre menu s'applique à la barre de menus du process courant. Sinon, Titre menu s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Référence

Nombre de menus.

Texte ligne menu

Texte ligne menu (menu; ligneMenu{; process}) → Alpha

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne de menu
process	Numérique	→	Numéro de référence de process
Résultat	Alpha	←	Libellé de la ligne de menu

Description

La commande Texte ligne menu retourne le libellé de la commande de menu dont les numéros de menu et de commande ont été passés dans menu et cmdMenu.

Si vous ne passez pas le paramètre process, Texte ligne menu est appliquée à la barre de menus du process courant. Sinon Texte ligne menu est appliquée à la barre de menus du process dont la référence est passée dans process.

Référence

CHANGER TEXTE LIGNE MENU.

CHANGER TEXTE LIGNE MENU

version 5

CHANGER TEXTE LIGNE MENU (menu; ligneMenu; texteLigne{; process})

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne de menu
texteLigne	Alpha	→	Nouveau libellé de la ligne de menu
process	Numérique	→	Numéro de référence de process

Description

La commande CHANGER TEXTE LIGNE MENU remplace le libellé de la ligne de menu, dont vous avez passé les numéros de menu et de ligne dans menu et ligneMenu, par le libellé que vous avez passé dans texteLigne.

Si vous omettez le paramètre process, CHANGER TEXTE LIGNE MENU s'applique à la barre de menus du process courant. Sinon, CHANGER TEXTE LIGNE MENU s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Référence

Texte ligne menu.

Style ligne menu (menu; ligneMenu{; process}) → Numérique

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne de menu
process	Numérique	→	Numéro de référence de process
Résultat	Numérique	←	Style courant de la ligne de menu

Description

La commande Style ligne menu retourne le style de police de la ligne de menu dont vous avez passé les numéros de menu et de commande dans menu et ligneMenu.

Si vous omettez le paramètre process, Style ligne menu s'applique à la barre de menus du process courant. Sinon, Style ligne menu s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Style ligne menu retourne une combinaison (une ou une somme) des constantes prédéfinies suivantes :

Constante	Type	Valeur
Standard	Entier long	0
Gras	Entier long	1
Italique	Entier long	2
Souligné	Entier long	4
Contours	Entier long	8
Ombré	Entier long	16
Condensé	Entier long	32
Etendu	Entier long	64

Note : Sous Windows, seuls les styles Standard ou une combinaison de Gras, Italique et Souligné sont disponibles.

Exemple
Si, par exemple, vous voulez tester si une ligne de menu est affichée en gras, vous écrivez :

```
⇒   Si ((Style ligne menu($vlMenu;$vlItem) & Gras)#0)
       ...
    Fin de si
```

Référence
CHANGER STYLE LIGNE MENU.

CHANGER STYLE LIGNE MENU

version 6.0

CHANGER STYLE LIGNE MENU (menu; ligneMenu; styleLigne{; process})

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne de menu
styleLigne	Numérique	→	Nouveau style de la ligne de menu
process	Numérique	→	Numéro de référence du process

Description

La commande CHANGER STYLE LIGNE MENU remplace le style de police de la ligne de menu, dont vous avez passé les numéros de menu et de commande dans menu et ligneMenu, par le style de police que vous avez passé dans styleLigne.

Si vous omettez le paramètre process, CHANGER STYLE LIGNE MENU s'applique à la barre de menus du process courant. Sinon, CHANGER STYLE LIGNE MENU s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Vous pouvez définir le style de l'élément dans le paramètre styleLigne. Vous passez une ou une combinaison des constantes prédéfinies suivantes :

Constante	Type	Valeur
Standard	Entier long	0
Gras	Entier long	1
Italique	Entier long	2
Souligné	Entier long	4
Contours	Entier long	8
Ombré	Entier long	16
Condensé	Entier long	32
Etendu	Entier long	64

Note : Sous Windows, seuls les styles Standard ou une combinaison de Gras, Italique et Souligné sont disponibles.

Référence

Style ligne menu.

Marque ligne menu (menu; ligneMenu{; process}) → Alpha

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne de menu
process	Numérique	→	Numéro de référence de process
Résultat	Alpha	←	Marque de ligne de menu courante

Description

La commande Marque ligne menu retourne la marque (ou "coche") de la ligne de menu dont vous avez passé le numéro de menu et de ligne dans menu et ligneMenu.

Si vous omettez le paramètre process, Marque ligne menu s'applique à la barre de menus du process courant. Sinon, Marque ligne menu s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Si la ligne de menu n'a pas de marque, Marque ligne menu retourne une chaîne vide.

Note : Pour plus d'informations sur les marques des lignes de menus sous MacOS et Windows, reportez-vous à la description de la commande MARQUER LIGNE MENU.

Exemple

L'exemple suivant inverse l'état marqué d'une ligne de menu :

```
⇒ MARQUER LIGNE MENU($vlMenu;$vlItem;Caractere(18)*Num
                                    (Marque ligne menu($vlMenu;$vlItem)=""))
```

Référence

MARQUER LIGNE MENU.

MARQUER LIGNE MENU (menu; ligneMenu; marque{; process})

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne de menu
marque	Alpha	→	Nouvelle marque de ligne de menu
process	Numérique	→	Numéro de référence du process

Description

La commande MARQUER LIGNE MENU remplace la marque (ou "coche") de la ligne de menu dont vous avez passé le numéro de menu et de ligne dans menu et ligneMenu par le premier caractère de la chaîne que vous avez passée dans marque (sous MacOS) ou par la coche standard (sous Windows).

Si vous omettez le paramètre process, MARQUER LIGNE MENU s'applique à la barre de menus du process courant. Sinon, MARQUER LIGNE MENU s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Si vous passez une chaîne vide dans marque, vous supprimez toute marque de la ligne de menu.
Sinon :
• Sous MacOS, le premier caractère de la chaîne devient la marque de la ligne de menu (généralement, le Caractere (18)), qui est la coche standard de MacOS, est utilisée.
• Sous Windows, la marque standard de Windows est associée au menu.

Exemple

Reportez-vous à l'exemple de la commande Marque ligne menu.

Référence

Marque ligne menu.

Raccourci clavier (menu; ligneMenu{; process}) → Numérique

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de la ligne de menu
process	Numérique	→	Numéro de référence de process
Résultat	Numérique	←	Code ASCII du raccourci de la ligne de menu

Description

La commande Raccourci clavier retourne le code ASCII de la touche **Ctrl** (sous Windows) ou **Commande** (MacOS) utilisée comme raccourci clavier pour la commande de menu dont les numéros de menu et de commande ont été passés dans menu et ligneMenu.

Si vous ne passez pas le paramètre process, Raccourci clavier est appliquée à la barre de menus du process courant. Sinon, Raccourci clavier est appliquée à la barre de menus du process dont la référence est passée dans process.

Si la commande de menus n'a pas d'équivalent clavier, Raccourci clavier retourne 0 (zéro).

Référence

CHANGER RACCOURCI CLAVIER.

CHANGER RACCOURCI CLAVIER (menu; cmdeMenu; touche{; process})

Paramètre	Type		Description
menu	Numérique	→	Numéro du menu
cmdeMenu	Numérique	→	Numéro de commande (ligne) de menu
touche	Numérique	→	Code ASCII du nouvel équivalent clavier
process	Numérique	→	Numéro de référence du process

Description

La commande CHANGER RACCOURCI CLAVIER remplace la touches **Ctrl** (Windows) ou **Commande** (Macintosh) utilisée dans le raccourci clavier de la commande de menu dont vous avez passé les numéros de menu et de commande dans menu et cmdeMenu, par le caractère dont vous avez passé le code ASCII dans touche.

Si vous ne passez pas le paramètre process, CHANGER RACCOURCI CLAVIER est appliquée à la barre de menus du process courant. Sinon, CHANGER RACCOURCI CLAVIER est appliquée à la barre de menus du process dont la référence est passée dans process.

Si vous passez 0 (zéro) dans touche, l'équivalent clavier de la commande de menu est supprimé.

Note : Afin de soigner l'ergonomie de votre interface, utilisez des caractères majuscules, des chiffres ou des symboles qui sont accessibles au clavier sans avoir recours à des "modifiers" autres que les touches Ctrl (Windows) et Commande (MacOS).

Référence

Raccourci clavier.

INACTIVER LIGNE MENU

version 5

INACTIVER LIGNE MENU (menu; ligneMenu{; process})

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne (commande) de menu
process	Numérique	→	Numéro de référence du process

Description

INACTIVER LIGNE MENU désactive la commande de menu dont vous avez passé le numéro de menu et de ligne dans menu et ligneMenu.

Si vous omettez le paramètre process, INACTIVER LIGNE MENU s'applique à la barre de menus du process courant. Sinon, INACTIVER LIGNE MENU s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Astuce : Pour activer/inactiver toutes les lignes de menus en une fois, passez 0 (zéro) dans ligneMenu.

Référence

ACTIVER LIGNE MENU.

ACTIVER LIGNE MENU

version 5

ACTIVER LIGNE MENU (menu; ligneMenu{; process})

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de commande (ligne) de menu
process	Numérique	→	Numéro de référence du process

Description

ACTIVER LIGNE MENU active la commande de menu dont vous avez passé le numéro de menu et de ligne dans menu et ligneMenu.

Si vous omettez le paramètre process, ACTIVER LIGNE MENU s'applique à la barre de menus du process courant. Sinon, ACTIVER LIGNE MENU s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Astuce : Pour activer/inactiver toutes les lignes de menus en une fois, passez 0 (zéro) dans ligneMenu.

Référence

INACTIVER LIGNE MENU.

AJOUTER LIGNE MENU (menu; libelléLigne{; process})

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
libelléLigne	Alpha	→	Libellé du ou des nouvelle(s) ligne(s) de menu
process	Numérique	→	Numéro de référence du process

Description

La commande AJOUTER LIGNE MENU ajoute une ou plusieurs ligne(s) de menu au menu dont vous avez passé le numéro dans menu.

Si vous omettez le paramètre process, AJOUTER LIGNE MENU s'applique à la barre de menus du process courant. Sinon, AJOUTER LIGNE MENU s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

AJOUTER LIGNE MENU vous permet d'ajouter une ou plusieurs lignes de menu en un seul appel.

Vous définissez les lignes à ajouter à l'aide du paramètre libelléLigne, de la manière suivante :
• Chaque ligne est séparée des autres par un point-virgule ";", "ligne1;ligne2;ligne3".
• Pour inactiver une ligne, placez une parenthèse ouvrante "(" dans son libellé.
• Pour définir une ligne de séparation, passez la valeur "(-" en tant que libellé.
• Pour définir le style de caractères d'une ligne, placez dans son libellé le symbole inférieur à "<" suivi d'une lettre. Voici les différents codes :

<B	Gras
<I	Italique
<U	Souligné
<O	Contours (MacOS seulement)
<S	Relief (MacOS seulement)

• Pour associer une coche à une ligne, insérez dans son libellé un point d'exclamation "!" suivi du caractère que vous voulez utiliser comme coche. Sous MacOS, le caractère est affiché ; sous Windows, une coche standard est affichée (quel que soit le caractère passé).

• Pour associer une icône à une ligne, insérez dans son libellé un accent circonflexe "^" suivi d'un caractère dont le code ASCII moins 48 plus 256 (ou plus 208) représente un numéro de ressource d'icône MacOS.
• Pour ajouter un raccourci clavier à une ligne, insérez dans son libellé une barre oblique "/" suivie du caractère de raccourci.

Note : Utilisez les menus avec un nombre "raisonnable" de lignes. Si, par exemple, vous voulez afficher plus de 50 lignes, envisagez plutôt d'employer une zone de défilement dans un formulaire.

Note : AJOUTER LIGNE MENU accepte un texte comportant jusqu'à 32 000 caractères alors que INSERER LIGNE MENU n'en accepte que 255.

Important : Les nouvelles lignes ne sont pas associées à des méthodes. Elles doivent donc être gérées par une méthode formulaire, à l'aide de la commande Menu choisi.

Exemple

L'exemple suivant ajoute les noms des polices de caractères disponibles dans un menu **Polices** qui, dans cet exemple, est le sixième menu de la barre de menus courante :

```
` Dans la méthode base Sur ouverture
` La liste des polices est chargée et les libellés construits
LISTE DES POLICES(<>asPolicesDispo)
<>atPoliceCmdMenus:=""
Boucle ($vlPolice;1;Taille tableau(<>asPolicesDispo))
   <>atPoliceCmdMenus:=<>atPoliceCmdMenus+";"+<>asPolicesDispo{$vlPolice}
Fin de boucle
```

Ensuite, dans toute méthode formulaire ou projet, vous pouvez écrire :

```
⇒ AJOUTER LIGNE MENU(6;<>atPoliceCmdMenus)
```

Référence

INSERER LIGNE MENU, SUPPRIMER LIGNE MENU.

INSERER LIGNE MENU (menu; aprèsLigne; texteLigne{; process})

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
aprèsLigne	Numérique	→	Numéro de commande de menu
libelléLigne	Alpha	→	Libellé de la ligne de menu à insérer
process	Numérique	→	Numéro de référence de process

Description

La commande INSERER LIGNE MENU insère de nouvelles lignes dans le menu dont le numéro est passé dans menu et les place après la ligne de menu dont le numéro est passé dans aprèsLigne.

Si vous ne passez pas le paramètre process, INSERER LIGNE MENU est appliquée à la barre de menus du process courant. Sinon, INSERER LIGNE MENU est appliquée à la barre de menus du process dont la référence est passée dans process.

INSERER LIGNE MENU vous permet d'insérer une ou plusieurs commandes de menus en une seule fois.

INSERER LIGNE MENU fonctionne comme AJOUTER LIGNE MENU, hormis pour les deux différences suivantes :
• INSERER LIGNE MENU vous permet d'insérer des commandes de menu partout dans le menu alors que AJOUTER LIGNE MENU les ajoute toujours à la fin du menu.
• Avec INSERER LIGNE MENU, le texte des commandes passé dans libelléLigne est limité à 255 caractères alors que pour la commande AJOUTER LIGNE MENU cette limitation est de 32 000 caractères.

Reportez-vous à la description de la commande AJOUTER LIGNE MENU pour plus de détails sur la définition des commandes de menus passée dans libelléLigne.

Important : Les nouvelles lignes n'ont pas de méthodes associées. Elles doivent donc être gérées à partir d'une méthode formulaire qui utilise la fonction Menu choisi.

Référence

AJOUTER LIGNE MENU.

SUPPRIMER LIGNE MENU

SUPPRIMER LIGNE MENU (menu; ligneMenu{; process})

Paramètre	Type		Description
menu	Numérique	→	Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne de menu
process	Numérique	→	Numéro de référence de process

Description

La commande SUPPRIMER LIGNE MENU supprime la ligne de menu dont vous avez passé les numéros de menu et de ligne dans menu et ligneMenu.

Si vous omettez le paramètre process, SUPPRIMER LIGNE MENU s'applique à la barre de menus du process courant. Sinon, SUPPRIMER LIGNE MENU s'applique à la barre de menus du process dont vous avez passé le numéro dans process.

Note : Pour soigner l'ergonomie de votre interface, ne laissez pas accessible un menu ne comportant aucune ligne.

Référence

AJOUTER LIGNE MENU, INSERER LIGNE MENU.

34

Messages

SUPPRIMER MESSAGES

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

Les commandes SUPPRIMER MESSAGES et LAISSER MESSAGES suppriment ou font apparaître les thermomètres de progression affichés par 4e Dimension lorsque le programme exécute des opérations de longue durée. Par défaut, les messages sont affichés.

Voici la liste des opérations du mode Utilisation qui peuvent provoquer l'affichage d'un thermomètre de progression : Application d'une formule, Génération d'un état semi-automatique, Export de données, Import de données, Tri, Génération d'un graphe, Recherche, Recherche par formulaire, Recherche par formule.

Le tableau suivant liste les commandes qui peuvent provoquer l'affichage du thermomètre de progression :

APPLIQUER A SELECTION
VALEURS DISTINCTES
ECRITURE DIF
ECRITURE SYLK
ECRITURE ASCII
GRAPHE SUR SELECTION
LECTURE DIF
CHERCHER
CHERCHER PAR EXEMPLE
CHERCHER PAR FORMULE DANS SELECTION
TRIER
LECTURE SYLK
LECTURE ASCII
JOINTURE
SELECTION RETOUR
REDUIRE SELECTION
ETAT
SCAN INDEX
CHERCHER PAR FORMULE
CHERCHER DANS SELECTION
TRIER PAR FORMULE

Exemple

L'exemple suivant supprime les thermomètres de progression avant d'effectuer un tri, puis les rétablit après l'opération :

```
⇒ SUPPRIMER MESSAGES
  TRIER ([Adresses]; [Adresses]CP; >; [Adresses]Nom2; >)
  LAISSER MESSAGES
```

LAISSER MESSAGES

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

Reportez-vous à la description de la commande SUPPRIMER MESSAGES.

ALERTE (message{; libelléBoutonOK})

Paramètre	Type		Description
message	Alpha	→	Message à afficher dans la boîte de dialogue d'alerte
libelléBoutonOK	Alpha	→	Libellé du bouton OK

Description

La commande ALERTE affiche une boîte de dialogue d'alerte composée d'une icône, d'un message et d'un bouton OK.

Vous passez le message à afficher dans le paramètre message. Ce message peut contenir jusqu'à 255 caractères. Si la longueur ou la largeur des caractères est trop importante par rapport à la zone du message, il sera tronqué.

Par défaut, le libellé du bouton OK est “OK”. Si vous voulez changer ce libellé, passez le nouveau libellé dans le paramètre optionnel libelléBoutonOK. Si nécessaire, la largeur du bouton OK est augmentée vers la gauche pour contenir ce nouveau libellé.

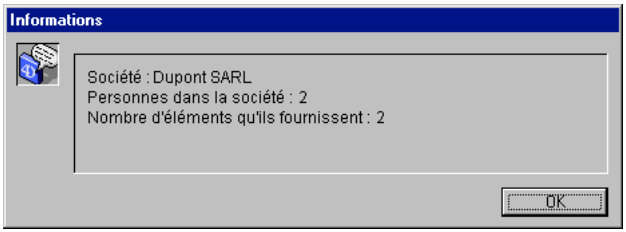
Note : N'appelez pas la commande ALERTE dans une méthode formulaire ou une méthode objet qui gère l'événement formulaire Sur activation ou Sur désactivation, car cela provoquerait une boucle sans fin.

Exemples

(1) L'exemple suivant appelle une boîte de dialogue d'alerte qui affiche des informations sur une société. Notez que le message contient des retours chariot (Caractere(13)) qui forcent le texte à passer sur la ligne suivante :

```
⇒ ALERTE("Société : "+[Sociétés]Nom+Caractere(13)+"Personnes dans la société : "+
                                    Chaine(Enregistrements trouves([Personne]))+
                                    Caractere(13)+"Nombre d'éléments qu'ils
                                fournissent : "+Chaine (Enregistrements trouves([Eléments])))
```

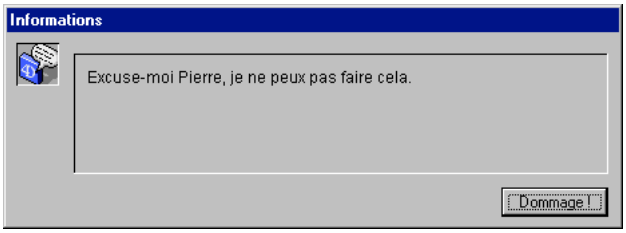
Voici la boîte de dialogue d'alerte affichée (sous Windows) par notre exemple :



(2) Voici un autre exemple :

⇒ **ALERTE**("Excuse-moi Pierre, je ne peux pas faire cela.";"Dommage !")

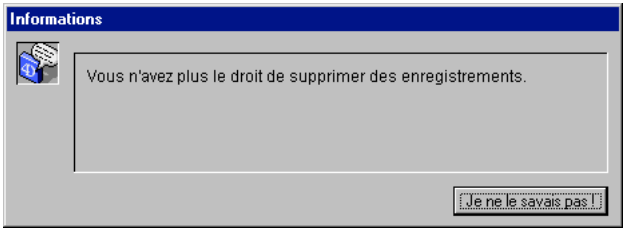
Cette instruction affichera (sous Windows) la boîte de dialogue d'alerte suivante :



(3) Voici un autre exemple :

⇒ **ALERTE**("Vous n'avez plus le droit de supprimer des enregistrements.";"Je ne le savais pas !")

Ce code affiche la boîte de dialogue d'alerte suivante :



Référence

CONFIRMER, Demander.

CONFIRMER (message{; libelléBoutonOK{; libelléBoutonAnn}})

Paramètre	Type		Description
message	Alpha	→	Message à afficher dans la boîte de dialogue de confirmation
libelléBoutonOK	Alpha	→	Libellé du bouton OK
libelléBoutonAnn	Alpha	→	Libellé du bouton Annuler

Description

La commande CONFIRMER affiche une boîte de dialogue de confirmation qui se compose d'une icône, d'un message, d'un bouton OK et d'un bouton Annuler.

Les boîtes de dialogue de confirmation ou d'alerte sont utilisées pour afficher des informations (comme des messages d'erreur) qui ne nécessitent pas d'informations en retour.

Vous passez le message à afficher dans le paramètre message. Ce message peut contenir jusqu'à 255 caractères. Si la longueur ou largeur des caractères est trop imporante par rapport à la zone d'affichage, le message sera tronqué.

Par défaut, le libellé du bouton OK est “OK” et le libellé du bouton Annuler est “Annuler”. Si vous voulez modifier le libellé de ces boutons, passez le nouveau libellé dans les paramètres optionnels libelléBoutonOK et libelléBouton Ann. Si nécessaire, les boutons sont agrandis vers la gauche en fonction de la taille des libellés que vous avez saisis.

Le bouton OK est le bouton par défaut. L'utilisateur peut cliquer sur le bouton OK ou appuyer sur la touche **Entrée** pour valider la boîte de dialogue, la variable système OK prend alors la valeur 1. L'utilisateur peut cliquer sur le bouton Annuler pour annuler la boîte de dialogue, la variable système OK prend alors la valeur 0.

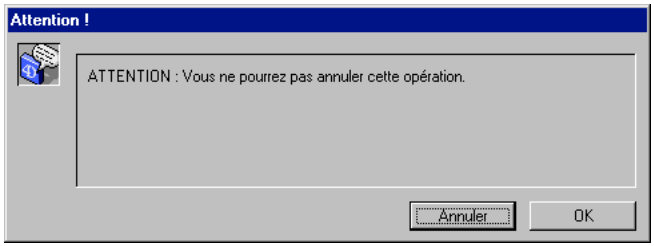
Conseil : N'appelez pas la commande CONFIRMER dans une méthode formulaire ou objet qui gère l'événement Sur activation ou Sur désactivation, car cela provoquerait une boucle sans fin.

Exemples

(1) L'exemple ci-dessous :

```
⇒    CONFIRMER("ATTENTION : Vous ne pourrez pas annuler cette opération.")
     Si (OK=1)
        TOUT SELECTIONNER([Vieilles choses])
        SUPPRIMER SELECTION([Vieilles choses])
     Sinon
        ALERTE ("Opération annulée.")
     Fin de si
```

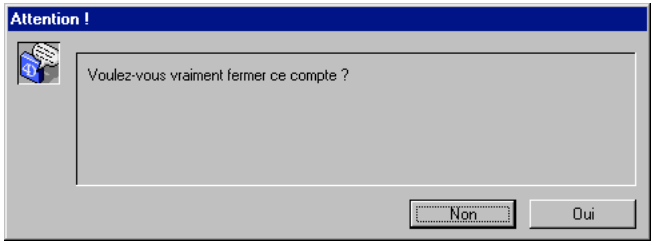
... provoquera l'affichage de la boîte de dialogue de confirmation suivante (sous Windows) :



(2) La ligne :

```
⇒    CONFIRMER("Voulez-vous vraiment fermer ce compte ?";"Oui";"Non")
```

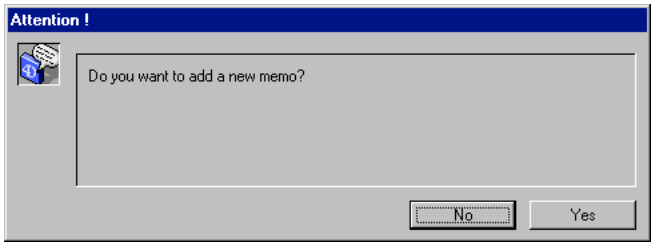
... provoquera l'affichage de la boîte de dialogue de confirmation suivante (sous Windows) :



(3) Vous développez une application 4D pour le marché international. Vous avez écrit une méthode projet qui retourne du texte traduit à partir d'une version française. Vous avez également rempli un tableau nommé asLocalizedUIMessages dans lequel vous stockez les mots les plus courants. Dans ce cas, la ligne :

⇒ **CONFIRMER**(*INTL Text* ("Voulez-vous ajouter un nouveau mémo ?");
asLocalizedUIMessages{kLoc_OUI};asLocalizedUIMessages{kLoc_NON})

... pourrait afficher la boîte de dialogue de confirmation (sous Windows) suivante :



(4) La ligne :

⇒ **CONFIRMER**("ATTENTION : Si vous poursuivez cette opération, des enregistrements
seront "+"modifiés irrémédiablement."+
Caractere(13)+"Que voulez-vous faire ?";
"Ne PAS continuer";"Continuer")

... provoque l'affichage de la boîte de dialogue de confirmation suivante (sous MacOS) :

Référence
ALERTE, Demander.

Demander (message{; réponseDéfaut{; titreBoutonOK{; titreBoutonAnn}}}) → Alpha

Paramètre	Type		Description
message	Alpha	→	Message à afficher dans la boîte de dialogue
réponseDéfaut	Alpha	→	Valeur par défaut dans la zone de saisie de texte
titreBoutonOK	Alpha	→	Libellé du bouton OK
titreBoutonAnn	Alpha	→	Libellé du bouton Annuler
Résultat	Alpha	←	Valeur saisie par l'utilisateur

Description

La fonction Demander affiche une boîte de dialogue de demande d'informations composée d'un message, d'une zone de saisie de texte, d'un bouton **OK** et d'un bouton **Annuler**.

Vous passez le message à afficher dans le paramètre message. Ce message peut contenir jusqu'à 255 caractères. Il peut cependant apparaître tronqué, en fonction de sa taille et de la largeur des caractères, s'il est supérieur à la capacité d'affichage de la zone de message.

Par défaut, le libellé du bouton OK est “OK” et celui du bouton Annuler est “Annuler”. Si vous voulez modifier ces libellés, passez d'autres valeurs dans les paramètres optionnels titreBoutonOK et titreBoutonAnn. Si nécessaire, les boutons sont agrandis vers la gauche, en fonction de la taille des libellés que vous avez saisis.

Le bouton OK est le bouton par défaut. L'utilisateur peut cliquer sur le bouton OK ou appuyer sur la touche Entrée pour valider la boîte de dialogue, mettant ainsi la variable système OK à 1. Il peut également cliquer sur le bouton Annuler pour annuler la boîte de dialogue, mettant ainsi la variable système OK à 0.

L'utilisateur peut taper des caractères dans la zone de saisie de texte. Pour définir une valeur par défaut, passez le texte par défaut dans le paramètre réponseDéfaut. Si l'utilisateur clique sur le bouton OK, Demander retourne le texte. Si l'utilisateur clique sur le bouton Annuler, Demander retourne une chaîne vide (""). Si la réponse doit être une valeur numérique ou une date, convertissez la chaîne retournée par Demander dans le type souhaité à l'aide des fonctions Num et Date.

Note : N'appelez pas la fonction Demander dans une méthode formulaire ou objet qui gère l'événement Sur activation ou Sur desactivation car cela provoquerait une boucle sans fin.

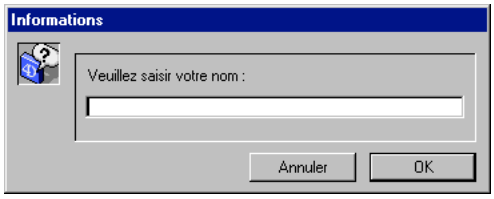
Conseil : Si vous voulez récupérer plusieurs informations de l'utilisateur, construisez un formulaire approprié et appelez-le avec la commande DIALOGUE, plutôt que d'afficher une succession de boîtes de dialogue du type Demander.

Exemples

(1) La ligne de code :

⇒ $vsAffiche := **Demander** ("Veuillez saisir votre nom :")

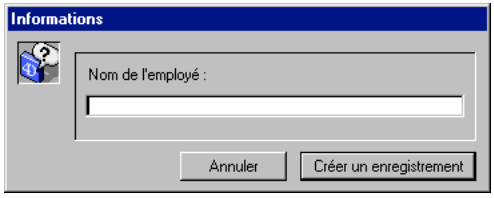
... provoquera l'affichage de la boîte de dialogue suivante :



(2) Le code suivant :

```
⇒ $vsAffiche:= Demander ("Nom de l'employé :";"";"Créer un enregistrement";"Annuler")
   Si (OK=1)
      AJOUTER ENREGISTREMENT([Employés])
         ` Note : $vsAffiche est alors copiée dans dans le champ [Employés]Nom
         ` lors de l'événement formulaire Sur chargement de la méthode formulaire
   Fin de si
```

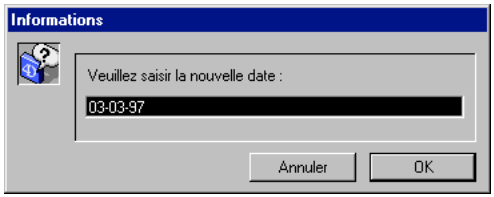
... provoquera l'affichage de la boîte de dialogue suivante :



(3) La ligne de code :

⇒ $vdAffiche := **Date** (**Demander** ("Veuillez saisir la nouvelle date :";**Chaine** (**Date du jour**)))

... provoquera l'affichage de la boîte de dialogue suivante :



Référence

ALERTE, CONFIRMER.

MESSAGE (message)

Paramètre	Type		Description
message	Alpha	→	Message à afficher

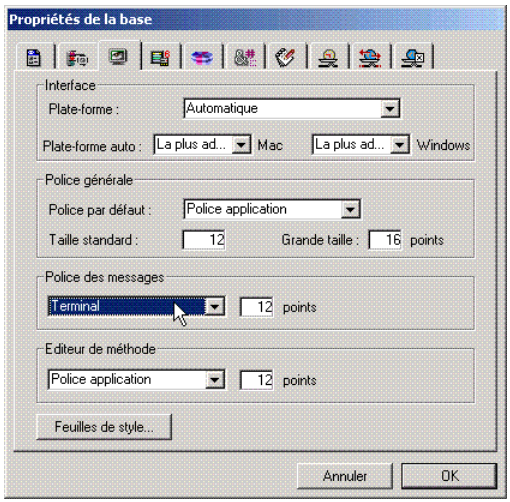
Description

La commande MESSAGE affiche message à l'écran dans une fenêtre spéciale de message qui est ouverte et refermée à chaque fois que vous l'appelez (à moins que vous ne travailliez dans une fenêtre préalablement ouverte par la commande Creer fenetre, cf. ci-dessous). Le message est temporaire et est effacé dès qu'un formulaire est affiché ou dès que l'exécution de la méthode est stoppée. Si une autre commande MESSAGE est exécutée, le précédent message est effacé.

MESSAGE est généralement utilisée pour informer l'utilisateur du déroulement d'une action.

Si une fenêtre a été ouverte par la commande Creer fenetre, tous les appels ultérieurs à la commande MESSAGE affichent les messages dans cette fenêtre. Cette fenêtre se comporte en quelque sorte comme un terminal :
• Chaque message successif n'efface pas le précédent, les messages se placent les uns à la suite des autres.
• Si un message est plus large que la fenêtre, 4e Dimension insère automatiquement un retour à la ligne.
• Si le message contient plus de lignes que ne peut en afficher la fenêtre, 4e Dimension fait automatiquement défiler le message dans la fenêtre.
• Si vous souhaitez contrôler les retours à la ligne, insérez vos propres retours chariot dans votre texte, à l'aide de Caractere(13).
• Vous pouvez appeler la commande POSITION MESSAGE pour afficher le texte à un emplacement particulier dans la fenêtre.
• Vous pouvez appeler la commande EFFACER FENETRE pour effacer le contenu de la fenêtre.
• La fenêtre est une fenêtre d'affichage statique : son contenu n'est pas redessiné lorsque d'autres fenêtres s'affichent par-dessus.

4e Dimension exploite les propriétés **Police des messages** et **Taille associée** pour afficher des messages. Vous définissez ces paramètres dans la boîte de dialogue "Propriétés de la base" :



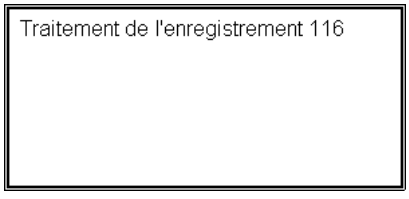
Vous pouvez fixer à votre convenance la police et la taille de la police (dans les limites du possible) pour les messages. Cependant, si vous combinez l'utilisation de MESSAGE et de POSITION MESSAGE, nous vous conseillons de choisir une police à espacement constant telle que **Terminal** sous Windows et **Monaco** sous MacOS.

Exemples

(1) L'exemple suivant traite une sélection d'enregistrements et appelle la commande MESSAGE pour informer l'utilisateur de la progression de l'opération :

```
      Boucle($vlEnregistrement;1;Enregistrements trouves([touteTable]))
⇒        MESSAGE ("Traitement de l'enregistrement "+Chaine($vlEnregistrement))
            ` Faire quelque chose avec l'enregistrement
         ENREGISTREMENT SUIVANT([touteTable])
      Fin de boucle
```

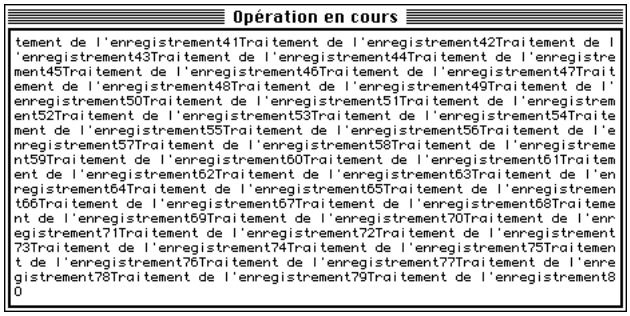
La fenêtre suivante s'affiche puis disparaît à chaque appel de MESSAGE :



(2) Afin d'éliminer le "clignotement" de la fenêtre, il est préférable, comme dans ce deuxième exemple, d'afficher les messages dans une fenêtre ouverte par l'intermédiaire de la commande Creer fenetre :

```
Creer fenetre(50;50;500;250;5;"Opération en cours")
Boucle($vlEnregistrement;1;Enregistrements trouves([touteTable]))
⇒    MESSAGE ("Traitement de l'enregistrement "+Chaine($vlEnregistrement))
        ` Faire quelque chose avec l'enregistrement
     ENREGISTREMENT SUIVANT([touteTable])
Fin de boucle
FERMER FENETRE
```

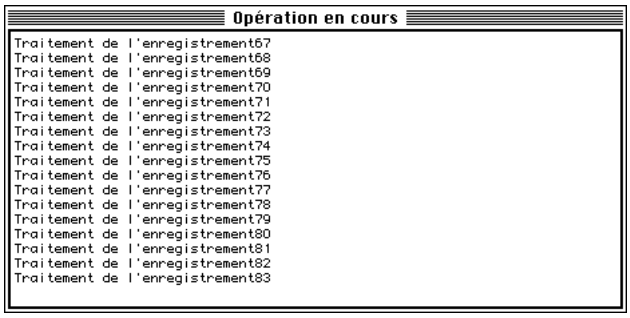
Le résultat est le suivant (sous MacOS) :



(3) En ajoutant un retour chariot, vous améliorez la présentation :

```
Creer fenetre(50;50;500;250;5;"Opération en cours")
Boucle($vlEnregistrement;1;Enregistrements trouves([touteTable]))
⇒    MESSAGE ("Traitement de l'enregistrement "+Chaine($vlEnregistrement)
                                                  +Caractere(13))
        ` Faire quelque chose avec l'enregistrement
     ENREGISTREMENT SUIVANT([touteTable])
Fin de boucle
FERMER FENETRE
```

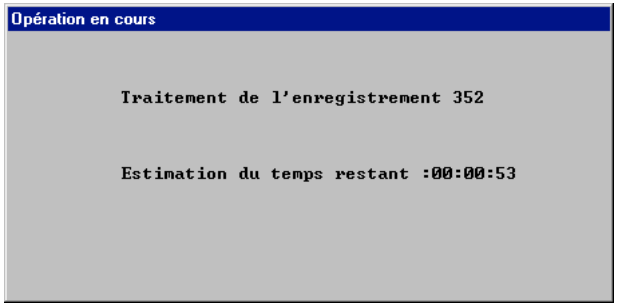
Voici le résultat (Sous MacOS) :



(4) A l'aide de la commande POSITION MESSAGE et de l'écriture de quelques lignes supplémentaires, la présentation s'améliore nettement :

```
Creer fenetre(50;50;500;250;5;"Opération en cours")
$vlNbEnregistrements:=Enregistrements trouves([touteTable])
$vhHeureDébut:=Heure courante
Boucle($vlEnregistrement;1;$vlNbEnregistrements)
   POSITION MESSAGE(5;2)
⇒  MESSAGE ("Traitement de l'enregistrement "+Chaine($vlNbEnregistrements)+
                                                    Caractere(13))
      ` Faire quelque chose avec les enregistrements
   ENREGISTREMENT SUIVANT([touteTable])
   POSITION MESSAGE(5;5)
   $vlReste:=(($vlNbEnregistrements/$vlEnregistrement)-1)*
                                          (Heure courante-$vhHeureDébut)
⇒  MESSAGE ("Estimation du temps restant : "+Chaine heure($vlReste))
Fin de boucle
FERMER FENETRE
```

Voici le résultat (sous Windows) :



Référence

Creer fenetre, EFFACER FENETRE, FERMER FENETRE, POSITION MESSAGE.

POSITION MESSAGE (x; y)

Paramètre	Type		Description
x	Numérique	→	Coordonnée x (horizontale) du curseur
y	Numérique	→	Coordonnée y (verticale) du curseur

Description

La commande POSITION MESSAGE est destinée à être utilisée conjointement avec la commande MESSAGE lorsque vous affichez des messages dans une fenêtre ouverte par la commande Creer fenetre.

La commande POSITION MESSAGE détermine l'emplacement du curseur d'insertion des caractères (ce curseur est invivible) : elle définit les coordonnées auxquelles le prochain message s'affichera à l'intérieur de la fenêtre.

L'angle supérieur gauche de la fenêtre représente les coordonnées 0,0. Le curseur est automatiquement positionné à 0,0 lorsqu'une fenêtre est créée ou après l'exécution de la commande EFFACER FENETRE.

Après que POSITION MESSAGE ait défini l'emplacement du curseur, la commande MESSAGE peut être appelée pour afficher des caractères dans la fenêtre.

Conseil : Pour contrôler parfaitement l'affichage des caractères avec les commandes POSITION MESSAGE et MESSAGE, utilisez des polices à espacement constant (par exemple Terminal sous Windows et Monaco sous MacOS). Dans ces polices, tous les caractères ont la même largeur. Reportez-vous à la description de la commande MESSAGE pour plus d'informations.

Exemples

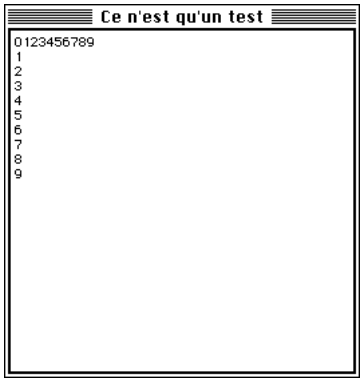
(1) Reportez-vous à l'exemple de la commande MESSAGE.

(2) Reportez-vous à l'exemple de la fonction Nombre de millisecondes.

(3) L'exemple ci-dessous :

```
CREER FENETRE(50;50;300;300;5;"Ce n'est qu'un test")
Boucle ($vlColonne;0;9)
⇒    POSITION MESSAGE($vlColonne;0)
     MESSAGE(Chaine($vlColonne))
Fin de boucle
Boucle ($vlLigne;0;9)
⇒    POSITION MESSAGE(0;$vlLigne)
     MESSAGE(Chaine($vlLigne))
Fin de boucle
$vhHeureDébut:=Heure courante
Repeter
Jusque ((Heure courante-$vhHeureDébut)>†00:00:30†)
```

... affiche la fenêtre suivante (sous MacOS) pendant 30 secondes :



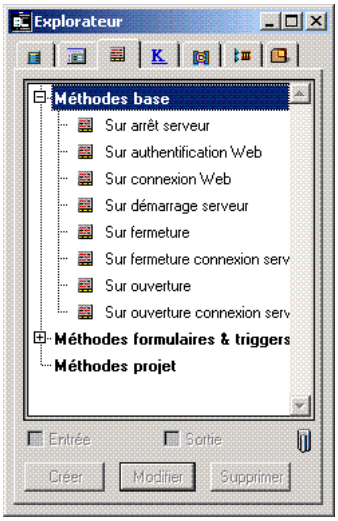
Référence

MESSAGE.

35

Méthodes base

Les méthodes base sont des méthodes automatiquement exécutées par 4D lors d'un événement affectant la session dans sa généralité.



Pour créer, ouvrir ou éditer une méthode base :

1. Ouvrez la fenêtre de l'**Explorateur**.
2. Sélectionnez l'onglet **Méthodes**.
3. Déployez le thème **Méthodes base**.
4. Double-cliquez sur la méthode.

ou bien :

4. Sélectionnez la méthode.
5. Cliquez sur le bouton **Modifier** ou appuyez sur la touche **Entrée** ou **Retour chariot**.

Vous éditez une méthode base de la même manière que n'importe quelle autre méthode.

Vous ne pouvez pas appeler une méthode base depuis une autre méthode. Les méthodes base sont automatiquement exécutées par 4D à certains moment de la session de travail. Le tableau suivant résume l'exécution des méthodes base :

Méthode base	4e Dimension	4D Server	4D Client
Sur ouverture	Oui, une fois	Non	Oui, une fois
Sur fermeture	Oui, une fois	Non	Oui, une fois
Sur authentification Web	Oui, plusieurs fois	Oui, plusieurs fois	Non
Sur connexion Web	Oui, plusieurs fois	Oui, plusieurs fois	Non
Sur démarrage serveur	Non	Oui, une fois	Non
Sur arrêt serveur	Non	Oui, une fois	Non
Sur ouverture connexion serveur	Non	Oui, plusieurs fois	Non
Sur fermeture connexion serveur	Non	Oui, plusieurs fois	Non

Pour plus de détails sur chaque méthode base, reportez-vous aux sections suivantes :

- Méthode base Sur ouverture
- Méthode base Sur fermeture
- Méthode base Sur authentification Web
- Méthode base Sur connexion Web
- Méthode base Sur démarrage serveur (cf. *Guide de référence* de 4D Server)
- Méthode base Sur arrêt serveur (cf. *Guide de référence* de 4D Server)
- Méthode base Sur ouverture connexion serveur (cf. *Guide de référence* de 4D Server)
- Méthode base Sur fermeture connexion serveur (cf. *Guide de référence* de 4D Server)

Référence

Méthodes.

La Méthode base Sur ouverture est exécutée une seule fois, au moment de l'ouverture de la base.

Les environnement 4D suivants sont concernés :
• 4e Dimension
• 4D Client (sur le poste client une fois que la connexion a été acceptée par 4D Server)
• 4D Runtime
• Application 4D compilée par 4D Compiler et fusionnée avec 4D Engine

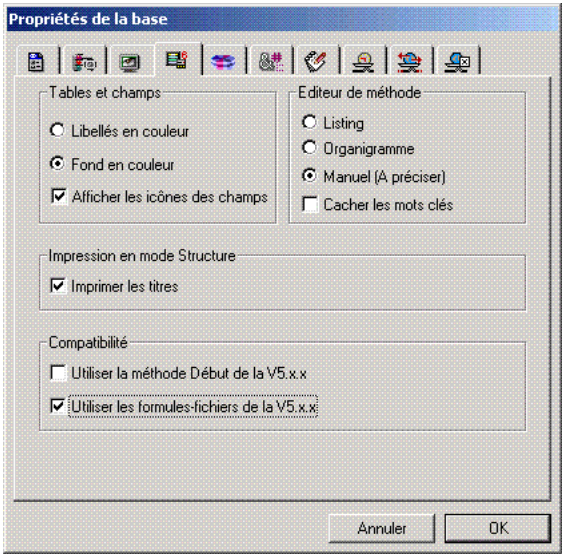
Note : La Méthode base Sur ouverture n'est PAS exécutée par 4D Server.

La méthode base Sur ouverture est exécutée automatiquement par 4D ; à la différence des méthodes projet, vous ne pouvez pas appeler cette méthode base vous-même. Pour exécuter des tâches depuis la méthode base Sur ouverture, vous pouvez, tout comme avec des méthodes projet, utiliser des sous-routines.

La méthode base Sur ouverture est l'emplacement idéal pour :
• Initialiser les variables interprocess que vous utiliserez pendant toute la session de travail.
• Démarrer automatiquement des process à l'ouverture de la base.
• Charger des préférences ou des paramétrages sauvegardés dans ce but lors de la session de travail précédente.
• Empêcher l'ouverture de la base si une condition n'est pas remplie (comme par exemple, une ressource système manquante) par l'appel explicite de la commande QUITTER 4D.
• Accomplir toute autre action que vous souhaitez exécuter automatiquement à chaque ouverture de la base.

Compatibilité avec les versions précédentes de 4D

Les méthodes base sont un nouveau type de méthode introduit dans la version 6 de 4D. Dans les versions précédentes, il n'y avait qu'une seule méthode (procédure) que 4D exécutait automatiquement à l'ouverture de la base. Cette procédure devait être nommée Debut (en version française) ou Startup (dans les versions américaine et internationale). Si vous avez converti une base version 5 en version 6, et si vous désirez tirer parti des possibilités de la nouvelle méthode base Sur ouverture, désélectionnez la propriété **Utiliser l'ancienne méthode Début** dans la boîte de dialogue Propriétés de la base. Cette propriété ne concerne que le choix entre Debut et la méthode base Sur ouverture. Si vous n'avez pas désélectionné cette propriété et que vous créez, par exemple, une méthode base Sur fermeture, celle-ci sera exécutée par 4D.



Exemple

Reportez-vous à l'exemple de la section Méthode base Sur fermeture.

Référence

Méthode base Sur fermeture, Méthodes, Présentation des méthodes base, QUITTER 4D.

La Méthode base Sur fermeture est appelée une fois lorsque vous quittez la base.

Les environnements 4D suivants sont concernés :
• 4e Dimension
• 4D Client (côté client)
• 4D Runtime
• Application 4D compilée par 4D Compiler et fusionnée avec 4D Engine

Note : La Méthode base Sur fermeture n'est PAS exécutée par 4D Server.

La méthode base Sur fermeture est exécutée automatiquement par 4D ; à la différence des méthodes projet, vous ne pouvez pas appeler cette méthode base vous-même. Pour exécuter des tâches depuis la méthode base Sur fermeture, vous pouvez, tout comme avec des méthodes projet, utiliser des sous-routines.

On sort de la base lorsque l'un des événements suivants se produit :
• L'utilisateur sélectionne la commande **Quitter** dans le menu **Fichier** en mode Structure ou Utilisation.
• Un appel à la commande QUITTER 4D a eu lieu.
• Un plug-in 4D a fait appel au point d'entrée QUITTER 4D.

Quel que soit le moyen par lequel la base a été quittée, 4D accomplit les actions qui suivent :

• S'il n'existe pas de méthode base Sur fermeture, 4D détruit chaque process un par un, sans distinction. Si un utilisateur est en train de saisir des données, les enregistrements ne seront pas sauvegardés.

• S'il existe une méthode base Sur fermeture, 4D démarre l'exécution de cette méthode dans un process local nouvellement créé. Vous pouvez ainsi utiliser cette méthode base pour informer d'autres process, via la communication interprocess, qu'ils doivent être fermés (en cas de saisie de données) ou stopper leur exécution. Notez que 4D quittera en tout état de cause — la méthode base Sur fermeture peut assurer le nettoyage et la fermeture de toutes les opérations que vous voulez, mais la fermeture de la base est inéluctable.

La Méthode base Sur fermeture est l'emplacement idéal pour :
• Stopper les process automatiquement démarrés à l'ouverture de la base.
• Sauvegarder (localement, sur disque) les préferences ou paramétrages devant être réutilisés lors de la prochaine session dans la méthode base Sur ouverture.
• Accomplir toute autre action que vous souhaitez exécuter automatiquement à chaque fermeture de la base.

Note: Rappelez-vous que le process créé pour la Méthode base Sur fermeture est un process client (local), qui n'existe donc pas sur le poste serveur. Par conséquent, si vous effectuez dans cette méthode base une recherche ou un tri, tout poste client qui tentera de quitter l'application restera "bloqué". Si vous avez besoin d'accéder aux données lorsque le client quitte, vous devez créer depuis cette méthode base un process global qui, lui, pourra accéder aux données. Dans ce cas toutefois, veillez à ce que ce process puisse terminer son exécution (par l'intermédaire de variables interprocess, par exemple) avant d'être stoppé par la Méthode base Sur fermeture.

Exemple

L'exemple ci-dessous liste les méthodes utilisées dans une base qui note les événements significatifs se produisant lors d'une session de travail. Les étapes sont écrites dans un document texte appelé “Journal”.

• La méthode base Sur ouverture initialise la variable interprocess ◊vbQuit4D, qui signale tous les process utilisés, qu'on sorte ou non de la base. Elle crée aussi le fichier journal, si il n'existe pas déjà.

```
   ` Méthode base Sur ouverture
C_TEXTE(◊vtIPMessage)
C_BOOLEEN(◊vbQuit4D)
◊vbQuit4D:=Faux

Si (Tester chemin acces("Journal") # Est un document)
   $vhDocRef:=Creer document("Journal")
   Si (OK=1)
      FERMER DOCUMENT($vhDocRef)
   Fin de si
Fin de si
ECRIRE JOURNAL ("Ouverture Session")
```

• La méthode projet ECRIRE JOURNAL, utilisée comme sous-routine par les autres méthodes, écrit l'information qu'elle reçoit dans le fichier journal :

```
   ` Méthode Projet ECRIRE JOURNAL
   ` ECRIRE JOURNAL ( Texte )
   ` ECRIRE JOURNAL ( Description Evenement )
C_TEXTE($1)
C_HEURE($vhDocRef)

Tant que (Semaphore("$Journal"))
   ENDORMIR PROCESS(Numero du process courant;1)
Fin tant que
$vhDocRef:=Ajouter a document("Journal")
Si (OK=1)
   INFORMATIONS PROCESS(Numero du process courant;$vsProcessNom;$vlEtat;
                                                   $vlTempsEcoule;$vbVisible)
   ENVOYER PAQUET($vhDocRef;Chaine(Date du jour)+Caractere(9)+
                                          Chaine(Date du jour)+Caractere(9)+
                                  Chaine(Numero du process courant)+Caractere(9)+
                                    $vsProcessNom+Caractere(9)+$1+Caractere(13))
   FERMER DOCUMENT($vhDocRef)
Fin de si
EFFACER SEMAPHORE("$Journal")
```

Notez que le document est ouvert et refermé à chaque fois. Notez aussi l'emploi d'un sémaphore comme “protection d'accès” au document — nous ne voulons pas que deux process essaient d'accéder au fichier journal en même temps.

• La méthode projet M_AJOUT_ENRG est exécutée lorsque la commande de menu **Ajouter enregistrement** est sélectionnée en mode Menus créés :

```
   ` Méthode Projet M_AJOUT_ENRG

CHANGER BARRE(1)
Repeter
   AJOUTER ENREGISTREMENT([Table1];*)
   Si (OK=1)
      ECRIRE JOURNAL("Ajout d'enregistrement #"+
                          Chaine(Numero enregistrement([Table1]))+" dans Table1")
   Fin de si
Jusque ((OK=0) | ◊vbQuit4D)
```

Cette méthode effectue une boucle jusqu'à ce que l'utilisateur annule la saisie de données ou que la base soit refermée.

• Le formulaire entrée de la [Table1] inclut le traitement des événements Sur appel extérieur. Ainsi, même si un process est en saisie de données, on en sort "en douceur", et l'utilisateur peut sauvegarder (ou non) la saisie en cours :

```
   ` Méthode formulaire [Table1];"Entrée"
Au cas ou
   : (Evenement formulaire=Sur appel extérieur)
      Si (◊vtIPMessage="QUITTER")
         CONFIRMER("Voulez-vous sauvegarder les modifications dans cet
                                                                  enregistrement ?")
         Si (OK=1)
            VALIDER
         Sinon
            NE PAS VALIDER
         Fin de si
      Fin de si
Fin de cas
```

• La méthode projet M_QUIT est exécutée lorsque la commande **Quitter** du menu **Fichier** en mode Menus créés est sélectionnée :

```
   ` Méthode Projet M_QUIT
$vlProcessID:=Nouveau process("ON_QUIT";32*1024;"$ON_QUIT")
```

Cette méthode utilise une astuce. Lorsque la commande QUITTER 4D est appelée, elle a un effet immédiat. En conséquence, le process dans lequel elle est appelée est placé en “mode arrêt”, jusqu'à ce que la base ait été effectivement refermée. Comme ce process peut être un des process dans lequel est effectuée la saisie de données, l'appel à QUITTER 4D est réalisé dans un process local qui n'est démarré que pour ce but. Voici la méthode ON_QUIT:

```
   ` Méthode projet ON_QUIT
CONFIRMER("Etes-vous certain de vouloir quitter ?")
Si (OK=1)
   ECRIRE JOURNAL ("Sortie de la base")
   QUITTER 4D
      ` QUITTER 4D a un effet immédiat. Aucune ligne de code n'est exécutée
                                                                  par la suite.
      ` ...
Fin de si
```

• Enfin, voici la méthode base Sur fermeture, qui signale à tous les process utilisateur qu'"il est temps de partir !". Elle met ◊vbQuit4D à Vrai et envoie des messages interprocess aux process utilisateur qui gèrent la saisie de données :

```
      ` Méthode base Sur fermeture
   ◊vbQuit4D:=Vrai
   Repeter
      $vbfini:=Vrai
      Boucle ($vlProcess;1;Nombre de process)
         INFORMATIONS PROCESS($vlProcess;$vsProcessNom;$vlEtat;$vlTempsEcoulé;
                                                                  $vbVisible)
         Si ((((($vsProcessNom="ML_@") | ($vsProcessNom="M_@"))) & ($vlEtat>=0))
            $vbDFini:=Faux
            ◊vtIPMessage:="QUITTER"
            PASSER AU PREMIER PLAN($vlProcess)
            APPELER PROCESS($vlProcess)
            $vhStart:=Heure courante
            Repeter
               ENDORMIR PROCESS(Numero du process courant;60)
            Jusque ((Statut du process($vlProcess)<0) | ((Heure courante-$vhStart)
                                                        >=?00:01:00?))
         Fin de si
      Fin de boucle
   Jusque ($vbDone)
   ECRIRE JOURNAL ("Fermeture de session")
```

Note : Les process dont les noms commencent par "ML_..." ou "M_..." sont démarrés par les commandes de menus pour lesquelles la propriété **Démarrer un process** a été sélectionnée. Dans cet exemple, ce sont les process démarrés suite à la sélection de la commande de menu **Ajouter enregistrement**.

Le test (Heure courante-$vhStart)>=?00:01:00? permet à la méthode base de sortir de la boucle "en attente de l'autre process", si l'autre process ne réagit pas pendant une minute.

• Voici un exemple type de fichier Journal produit par la base :

2/6/97	15:47:25	1	Process principal	Ouverture de Session
2/6/97	15:55:43	5	ML_1	Ajout de fiche #23 dans Table1
2/6/97	15:55:46	5	ML_1	Ajout de fiche #24 dans Table1
2/6/97	15:55:54	6	$On_QUIT	Sortie de la base
2/6/97	15:55:58	7	$xx	Fermeture de session

Note : $xx est le nom du process local démarré par 4D pour exécuter la méthode base Sur fermeture.

Référence

Méthode base Sur ouverture, QUITTER 4D.

36

Opérateurs

Les opérateurs sont des symboles permettant d'effectuer des opérations sur des expressions. Les opérateurs peuvent effectuer des calculs sur des nombres, des dates et des heures. Ils effectuent aussi des opérations logiques sur les chaînes, les booléens et des expressions logiques ainsi que des opérations spéciales sur des images. Ils combinent des expressions simples pour générer de nouvelles expressions.

Priorité

L'ordre dans lequel une expression est évaluée s'appelle la priorité. 4e Dimension applique strictement une règle de priorité de gauche à droite. **L'ordre algébrique n'est pas appliqué**. Par exemple :

```
3 + 4 * 5
```

retourne 35 car l'expression est évaluée comme 3 + 4, qui donne 7, multiplié par 5, ce qui donne 35.

Les parenthèses doivent être utilisées pour forcer l'ordre de calcul en fonction de vos besoins. Par exemple :

```
3 + (4 * 5)
```

retourne 23 car l'expression (4 * 5) est évaluée en premier lieu. Le résultat (20) est alors ajouté à 3, ce qui donne le résultat final 23.

Des parenthèses peuvent être incluses dans d'autres parenthèses. Assurez-vous qu'il y ait une parenthèse fermante pour chaque parenthèse ouverte. Une parenthèse manquante ou placée à un mauvais endroit peut soit donner un résultat erroné, soit renvoyer une expression invalide. De plus, si vous avez l'intention de compiler vos applications avec 4D Compiler, vous devez vous assurer d'une bonne utilisation des parenthèses. Le compilateur interprètera toute parenthèse manquante ou superflue comme une erreur de syntaxe.

L'opérateur d'assignation

L'opérateur d'assignation := se distingue des autres opérateurs. Au lieu de combiner des expressions en une seule, l'opérateur d'assignation copie la valeur de l'expression située à sa droite dans la variable ou le champ qui se trouve à sa gauche.

Par exemple, la ligne suivante place la valeur 4 (le nombre de caractères présents dans le mot Pont) dans la variable maVar, qui prend alors le type numérique.

```
maVar := Longueur ("Pont")
```

Important : Ne confondez pas l'opérateur d'assignation := avec l'opérateur de comparaison d'égalité =.

Les autres opérateurs proposés par le langage de 4D sont décrits dans les sections suivantes :

Opérateurs sur les chaînes
Référez-vous à la section Opérateurs sur les chaînes.

Opérateurs numériques
Référez-vous à la section Opérateurs numériques.

Opérateurs sur les dates
Référez-vous à la section Opérateurs sur les dates.

Opérateurs sur les heures
Référez-vous à la section Opérateurs sur les heures.

Comparateurs
Référez-vous à la section Opérateurs de comparaison.

Opérateurs logiques
Référez-vous à la section Opérateurs logiques.

Opérateurs sur les images
Référez-vous à la section Opérateurs sur les images.

Opérateurs sur les bits
Référez-vous à la section Opérateurs sur les bits.

Opérateurs sur les chaînes

Une expression qui utilise un opérateur sur les chaînes alphanumériques retourne une chaîne. Le tableau suivant décrit les opérateurs sur les chaînes :

Opération	Syntaxe	Retourne	Expression	Valeur
Concaténation	Chaîne + Chaîne	Chaîne	"abc" + "def"	"abcdef"
Répétition	Chaîne * Nombre	Chaîne	"ab" * 3	"ababab"

Référence

Opérateurs, Opérateurs de comparaison, Opérateurs logiques, Opérateurs numériques, Opérateurs sur les bits, Opérateurs sur les dates, Opérateurs sur les heures, Opérateurs sur les images.

Opérateurs numériques

Une expression qui utilise un opérateur numérique retourne une valeur numérique. Le tableau suivant décrit les opérateurs numériques :

Opération	Syntaxe	Retourne	Expression	Valeur
Addition	Nombre + Nombre	Nombre	2 + 3	5
Soustraction	Nombre – Nombre	Nombre	3 – 2	1
Multiplication	Nombre * Nombre	Nombre	5 * 2	10
Division	Nombre /Nombre	Nombre	5 / 2	2.5
Division entière	Nombre \ Nombre	Nombre	5 \ 2	2
Modulo	Nombre % Nombre	Nombre	5 % 2	1
Exponentiation	Nombre ^ Nombre	Nombre	2 ^ 3	8

L'opérateur modulo % divise le premier nombre par le second et retourne le reste de la division entière. Voici quelques exemples :

- 10 % 2 retourne 0 car la division de 10 par 2 ne donne pas de reste.
- 10 % 3 retourne 1 car le reste est 1.
- 10,5 % 2 retourne 0 car le reste n'est pas un nombre entier.

ATTENTION :

- L'opérateur modulo % retourne des valeurs significatives avec des nombres appartenant à la catégorie des entiers longs (de –2^31 à +2^31 moins 1). Pour calculer le modulo de nombres qui ne sont pas dans cet intervalle, utilisez la fonction Modulo.
- L'opérateur division entière \ retourne des valeurs significatives avec des nombres entiers uniquement.

Référence

Opérateurs, Opérateurs de comparaison, Opérateurs logiques, Opérateurs sur les bits, Opérateurs sur les chaînes, Opérateurs sur les dates, Opérateurs sur les heures, Opérateurs sur les images.

Opérateurs sur les dates

Une expression qui utilise un opérateur sur les dates retourne une date ou une valeur numérique, suivant l'opération effectuée. Toutes les opérations sur les dates retournent des valeurs exactes, tenant compte en particulier des années bissextiles. Le tableau suivant décrit les opérateurs sur les dates :

Opération	Syntaxe	Retourne	Expression	Valeur
Différence	Date – Date	Nombre	!20/1/97! – !1/1/97!	19
Addition	Date + Nombre	Date	!20/1/97! + 9	!29/1/97!
Soustraction	Date – Nombre	Date	!20/1/97! – 9	!11/1/97!

Référence

Opérateurs, Opérateurs de comparaison, Opérateurs logiques, Opérateurs numériques, Opérateurs sur les bits, Opérateurs sur les chaînes, Opérateurs sur les heures, Opérateurs sur les images.

Une expression qui utilise un opérateur sur les heures retourne une heure ou une valeur numérique, suivant l'opération effectuée. Le tableau suivant décrit les opérateurs sur les heures :

Opération	Syntaxe	Retourne	Expression	Valeur
Addition	Heure + Heure	Heure	?02:03:04? + ?01:02:03?	?03:05:07?
Soustraction	Heure – Heure	Heure	?02:03:04? – ?01:02:03?	?01:01:01?
Addition	Heure + Nombre	Nombre	?02:03:04? + 65	7449
Soustraction	Heure – Nombre	Nombre	?02:03:04? – 65	7319
Multiplication	Heure * Nombre	Nombre	?02:03:04? * 2	14768
Division	Heure / Nombre	Nombre	?02:03:04? / 2	3692
Division entière	Heure \ Nombre	Nombre	?02:03:04? \ 2	3692
Modulo	Heure % Nombre	Nombre	?02:03:04? % 2	0

Astuces

(1) Pour obtenir une expression de type heure à partir d'une expression qui combine une heure avec un chiffre, utilisez les fonctions Heure et Chaine heure. Par exemple :

```
   ` La ligne suivante assigne à la variable $vlSecondes le nombre de secondes qui,
   ` dans une heure à partir de maintenant, se seront écoulées depuis minuit
$vlSecondes:=Heure courante+3600

   ` La ligne suivante assigne à la variable $vhBientôt l'heure qu'il sera dans une heure
$vhBientôt:=Heure(Chaine heure(Heure courante+3600))
```

La seconde ligne peut également être écrite de la façon suivante :

```
   ` La ligne suivante affecte dans la variable $vhBientôt l'heure qu'il sera dans une
   ` heure
$vhBientôt:=Heure courante+?00:01:00?
```

Vous pouvez utiliser cette astuce si, lors du développement de votre application, vous vous retrouvez dans la situation où un délai exprimé en secondes doit être ajouté à une valeur de type heure disponible en tant que valeur numérique.

(2) Il faut parfois convertir une expression heure en expression numérique. Par exemple, vous ouvrez un document sur disque à l'aide de la fonction Ouvrir document, qui retourne un numéro de référence de document (DocRef) qui est une expression de type heure. Vous pouvez passer DocRef à une routine de plug-in 4D qui attend une valeur numérique comme numéro de référence de document. Dans ce cas, ajoutez 0 (zéro) à l'heure pour obtenir une valeur numérique, sans la modifier. Par exemple :

```
    ` Sélectionner et ouvrir un document
$vhDocRef:=Ouvrir document("")
Si (OK=1)
      ` Passez l'expression heure DocRef en tant qu'expression numérique à
      ` une routine d'extension 4D
   faire quelque chose (0+$vhDocRef)
Fin de si
```

Référence

Opérateurs, Opérateurs de comparaison, Opérateurs logiques, Opérateurs numériques, Opérateurs sur les bits, Opérateurs sur les chaînes, Opérateurs sur les dates, Opérateurs sur les images.

Opérateurs de comparaison

Les tableaux suivants décrivent les opérateurs de comparaison. Ces opérateurs peuvent être appliqués aux expressions de type chaîne, numérique, date, heure et pointeur (il n'est donc pas possible de les utiliser avec des expressions de type tableau, image ou BLOB).
Une expression qui utilise un opérateur de comparaison retourne une valeur booléenne, soit VRAI soit FAUX.

Comparaisons de chaînes

Opération	Syntaxe	Retourne	Expression	Valeur
Egalité	Chaîne = Chaîne	Booléen	"abc" = "abc" "abc" = "abd"	Vrai Faux
Inégalité	Chaîne # Chaîne	Booléen	"abc" # "abd" "abc" # "abc"	Vrai Faux
Supérieur à	Chaîne > Chaîne	Booléen	"abd" > "abc" "abc" > "abc"	Vrai Faux
Inférieur à	Chaîne < Chaîne	Booléen	"abc" < "abd" "abc" < "abc"	Vrai Faux
Supérieur ou égal à	Chaîne >= Chaîne	Booléen	"abd" >= "abc" "abc" >= "abd"	Vrai Faux
Inférieur ou égal à	Chaîne <= Chaîne	Booléen	"abc" <= "abd" "abd" <= "abc"	Vrai Faux

Comparaisons de numériques

Opération	Syntaxe	Retourne	Expression	Valeur
Egalité	Nombre = Nombre	Booléen	10 = 10 10 = 11	Vrai Faux
Inégalité	Nombre # Nombre	Booléen	10 #11 10 # 10	Vrai Faux
Supérieur à	Nombre > Nombre	Booléen	11 > 10 10 > 11	Vrai Faux
Inférieur à	Nombre < Nombre	Booléen	10 < 11 11 < 10	Vrai Faux
Supérieur ou égal à	Nombre >= Nombre	Booléen	11 >= 10 10 >= 11	Vrai Faux
Inférieur ou égal à	Nombre <= Nombre	Booléen	10 <= 11 11 <= 10	Vrai Faux

Comparaisons de dates

Opération	Syntaxe	Retourne	Expression	Valeur
Egalité	Date = Date	Booléen	!1/1/97! =!1/1/97! !20/1/97! =!1/1/97!	Vrai Faux
Inégalité	Date # Date	Booléen	!20/1/97! # !1/1/97! !1/1/97! # !1/1/97!	Vrai Faux
Supérieur à	Date > Date	Booléen	!20/1/97! > !1/1/97! !1/1/97! > !1/1/97!	Vrai Faux
Inférieur à	Date < Date	Booléen	!1/1/97! < !20/1/97! !1/1/97! < !1/1/97!	Vrai Faux
Supérieur ou égal à	Date >= Date	Booléen	!20/1/97! >=!1/1/97! !1/1/97!>=!20/1/97!	Vrai Faux
Inférieur ou égal à	Date <= Date	Booléen	!1/1/97!<=!20/1/97! !20/1/97!<=!1/1/97!	Vrai Faux

Comparaisons d'heures

Opération	Syntaxe	Retourne	Expression	Valeur
Egalité	Heure = Heure	Booléen	?01:02:03? = ?01:02:03? ?01:02:03? = ?01:02:04?	Vrai Faux
Inégalité	Heure # Heure	Booléen	?01:02:03? # ?01:02:04? ?01:02:03? # ?01:02:03?	Vrai Faux
Supérieur à	Heure > Heure	Booléen	?01:02:04? > ?01:02:03? ?01:02:03? > ?01:02:03?	Vrai Faux
Inférieur à	Heure < Heure	Booléen	?01:02:03? < ?01:02:04? ?01:02:03? < ?01:02:03?	Vrai Faux
Supérieur ou égal à	Heure >= Heure	Booléen	?01:02:03? >=?01:02:03? ?01:02:03? >=?01:02:04?	Vrai Faux
Inférieur ou égal à	Heure <= Heure	Booléen	?01:02:03? <=?01:02:03? ?01:02:04? <=?01:02:03?	Vrai Faux

Comparaisons de pointeurs

Avec :

```
   ` vPtrA et vPtrB pointent sur le même objet
vPtrA:=->unObjet
vPtrB:=->unObjet
   ` vPtrC pointe sur un autre objet
vPtrC:=->autreObjet
```

Opération	Syntaxe	Retourne	Expression	Valeur
Egalité	Pointeur = Pointeur	Booléen	vPtrA = vPtrB	Vrai
			vPtrA = vPtrC	Faux
Inégalité	Pointeur # Pointeur	Booléen	vPtrA # vPtrC	Vrai
			vPtrA # vPtrB	Faux

Remarques sur les comparaisons de chaînes

Voici quelques informations supplémentaires sur les comparaisons d'alphanumériques :

• Les chaînes sont toujours comparées caractère par caractère.

• Lors d'une comparaison de chaînes, 4e Dimension ne tient pas compte de la casse des caractères ; par exemple, "a"="A" retourne VRAI. Pour savoir si des caractères sont en majuscules ou en minuscules, vous devez comparer leurs codes ASCII. Par exemple, l'expression suivante retourne FAUX :

```
Code ascii ("A") = Code ascii ("a") ` 65 n'est pas égal à 97
```

• Lors d'une comparaison de chaînes, les caractères diacritiques sont comparés à l'aide de la table de comparaison des caractères de votre machine. Par exemple, les expressions suivantes retournent VRAI :

```
"n" = "ñ"
"n" = "Ñ"
"A"="å"
   ` etc
```

• Le joker (@) peut être utilisé dans toute comparaison de chaînes. Il remplace un ou plusieurs caractères. Ainsi, par exemple, l'expression suivante est évaluée à VRAI :

```
"abcdefghij" = "abc@"
```

Le joker doit être utilisé dans le second opérande (la chaîne qui se trouve à droite de l'opérateur). L'expression suivante est évaluée à FAUX car le joker est alors considéré en tant que caractère :

```
"abc@" = "abcdefghij"
```

Le joker signifie “un ou plusieurs caractères sinon rien”. Les expressions suivantes sont évaluées à VRAI :

```
"abcdefghij" = "abcdefghij@"
"abcdefghij" = "@abcdefghij"
"abcdefghij" = "abcd@efghij"
"abcdefghij" = "@abcdefghij@"
"abcdefghij" = "@abcde@fghij@"
```

En revanche, dans tous les cas, lorsque deux jokers consécutifs sont placés dans une comparaison de chaînes, celle-ci sera évaluée à FAUX. L'expression suivante est à FAUX :

```
"abcdefghij" = "abc@@fg"
```

Astuce

Si vous récupérez une chaîne lors de la saisie de données, elle peut contenir des jokers. Puisque ce caractère est utilisé dans la comparaison de chaînes, vous ne pouvez pas le traiter comme un autre caractère. Considérons l'exemple suivant :

```
$vaValeur:=Demander("Saisissez la valeur à chercher :")
Si (OK=1)
   CHERCHER ([Clients];[Clients]Nom=$vaValeur+"@")
Fin de si
```

Une valeur est demandée à l'aide de la fonction Demander. Ensuite, cette valeur est utilisée pour une recherche du type “commence par”. Puisque deux jokers consécutifs, comme décrit ci-dessus, provoquent un résultat FAUX, le joker doit être ajouté à la fin de la chaîne si et seulement si le dernier caractère n'est pas déjà un joker. Voici donc la nouvelle version de l'exemple :

```
$vaValeur:=Demander("Saisissez la valeur à chercher :")
Si (OK=1)
   Si (Code ascii($vaValeur≤Longueur($vaValeur)≥)#64)
      $vaValeur:=$vaValeur+"@"
   Fin de si
   CHERCHER ([Clients];[Clients]Nom=$vaValeur)
Fin de si
```

Notez l'utilisation de la fonction Code ascii car l'expression suivante (si $vaValeur n'est pas vide) retourne toujours VRAI :

```
$vaValeur≤Longueur($vaValeur)≥="@"
```

Toujours avec cet exemple, la chaîne saisie dans la boîte de dialogue peut contenir plusieurs jokers et mêmes des chaînes telles que "@@D@OE@@@". Le code suivant effacera tous les jokers présents dans la chaîne :

```
   ` Méthode Pas de jokers
   ` Pas de jokers ( Chaîne ) -> Chaîne
   ` Pas de jokers ( Toute chaîne ) -> Chaîne sans joker (@)
$0:=""
Boucle ($vlChar;1;Longueur($1))
   Si (Code ascii($1≤$vlChar≥)#64)
      $0:=$0+$1≤$vlChar≥
   Fin de si
Fin de boucle
```

Autrement dit, cette méthode projet fait la même chose que la fonction Remplacer chaine. Cependant, elle est nécessaire car elle traite les codes ASCII. En effet, l'expression suivante retournera toujours une chaîne vide :

```
Remplacer chaine($vsValeur;"@";"") ` Tous les caractères sont effacés
```

L'exemple devient donc :

```
$vaValeur:=Demander("Saisissez la valeur à chercher : ")
Si (OK=1)
   CHERCHER ([Clients];[Clients]Nom=Pas de jokers ($vaValeur)+"@")
Fin de si
```

Avec ce code, la recherche sera toujours de type “commence par”, quelle que soit la chaîne saisie par l'utilisateur.

Référence

Opérateurs, Opérateurs logiques, Opérateurs numériques, Opérateurs sur les bits, Opérateurs sur les dates, Opérateurs sur les heures, Opérateurs sur les images.

4e Dimension supporte deux opérateurs logiques : l'opérateur d'intersection (ET) et l'opérateur de réunion inclusive (OU). Ces deux opérateurs ne s'appliquent qu'aux expressions booléennes. Le ET logique retourne VRAI si les deux expressions sont VRAIES. Le OU logique retourne VRAI si au moins une des expressions est VRAIE.

4e Dimension vous permet également d'exploiter les fonctions booléennes Vrai, Faux et Non. Pour plus d'informations, reportez-vous aux descriptions de ces commandes.

Le tableau suivant décrit les opérateurs logiques :

Opération	Syntaxe	Retourne	Expression	Valeur
ET	Booléen & Booléen	Booléen	("A" = "A") & (15 # 3)	Vrai
			("A" = "B") & (15 # 3)	Faux
			("A" = "B") & (15 = 3)	Faux
OU	Booléen | Booléen	Booléen	("A" = "A") | (15 # 3)	Vrai
			("A" = "B") | (15 # 3)	Vrai
			("A" = "B") | (15 = 3)	Faux

Voici la "table de vérité" pour l'opérateur logique "ET" :

Expr1	Expr2	Expr1 & Expr2
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

Voici la "table de vérité" pour l'opérateur logique "OU" :

Expr1	Expr2	Expr1 | Expr2
Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

Astuce

Si vous devez calculer une réunion exclusive (le "OU" exclusif) entre Expr1 et Expr2, écrivez :

(Expr1 | Expr2) & **Non**(Expr1 & Expr2)

Référence

Opérateurs, Opérateurs de comparaison, Opérateurs numériques, Opérateurs sur les bits, Opérateurs sur les chaînes, Opérateurs sur les dates, Opérateurs sur les heures, Opérateurs sur les images.

Opérateurs sur les images

Le tableau suivant décrit les opérateurs que vous pouvez utiliser avec 4e Dimension sur les images. Une expression qui utilise un opérateur sur les images retourne toujours une image.

Opération	Syntaxe	Action
Concaténation horizontale	Image1 + Image2	Place Image2 à la droite d'Image1
Concaténation verticale	Image1 / Image2	Place Image2 au-dessous d'Image1
Superposition exclusive	Image1 & Image2	Effectue un OU exclusif entre Image1 et Image2
Superposition inclusive	Image1 | Image2	Effectue un OU inclusif entre Image1 et Image2
Déplacement horizontal	Image + Nombre	Déplace Image horizontalement d'un nombre de pixels égal à Nombre
Déplacement vertical	Image / Nombre	Déplace Image verticalement d'un nombre de pixels égal à Nombre
Redimensionnement	Image * Nombre	Redimensionne Image au pourcentage Nombre
Extension horizontale	Image *+ Nombre	Redimensionne Image horizontalement au pourcentage Nombre
Extension verticale	Image */ Nombre	Redimensionne Image verticalement au pourcentage Nombre

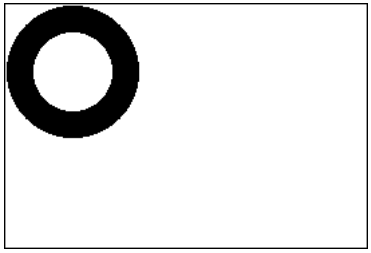
Les deux opérateurs & et | retournent toujours une image de type bitmap, quel que soit le type des deux images source. La raison en est que 4D dessine d'abord les images en mémoire en tant que bitmaps et calcule l'image résultante appliquant l'opérateur OU sur chaque pixel du bitmap.

Les autres opérateurs sur les images retournent des images vectorielles si les deux images source sont elles aussi vectorielles (rappelez-vous qu'une image imprimée avec le format d'affichage Sur fond est imprimée en tant que bitmap).

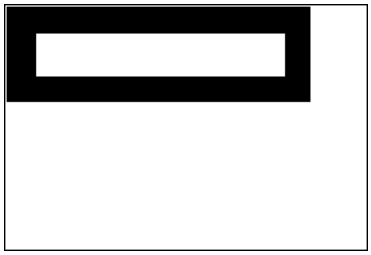
Exemples

Toutes les images qui sont affichées utilisent le format d'affichage Image sur fond.

Voici l'image cercle :



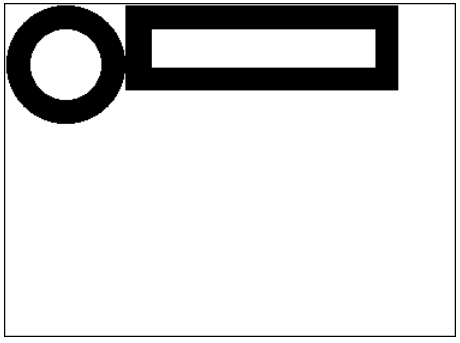
Voici l'image rectangle :



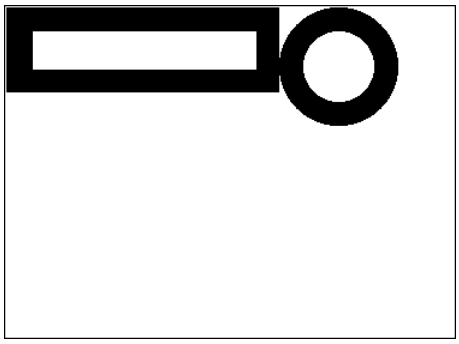
Dans les exemples ci-dessous, chaque expression est suivie de sa représentation graphique.

• Concaténation horizontale

cercle + rectangle ` Placer le rectangle à droite du cercle

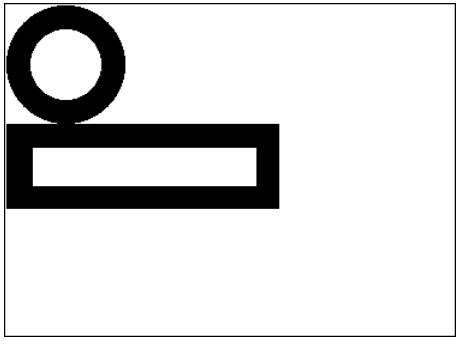


rectangle + cercle ` Placer le cercle à droite du rectangle

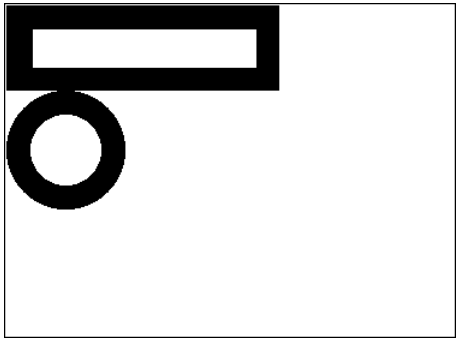


• Concaténation verticale

cercle / rectangle ` Placer le rectangle sous cercle

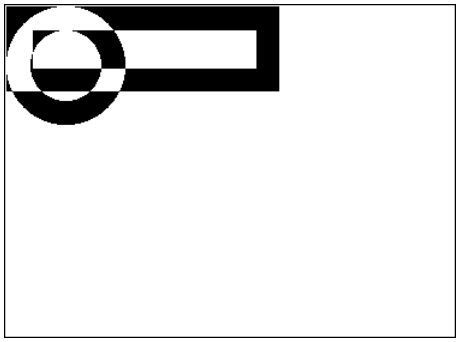


rectangle / cercle ` Placer le cercle sous le rectangle



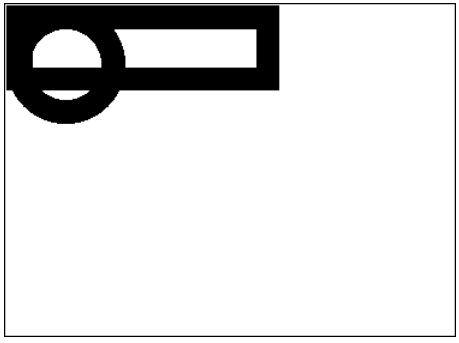
• Superposition exclusive (OU exclusif)

cercle & rectangle ` Exclusif OU des deux images



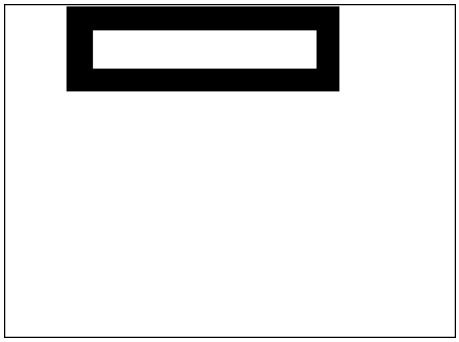
• Superposition inclusive (OU inclusif)

cercle | rectangle ` Inclusif OU des deux images

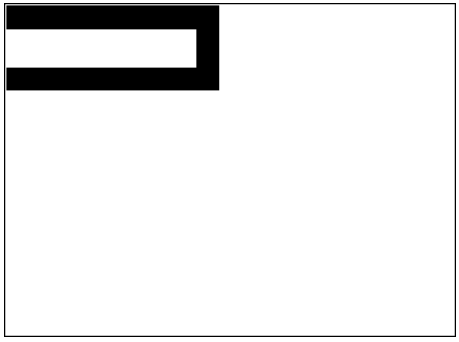


• Déplacement horizontal

rectangle + 50 ` Déplacer le rectangle 50 pixels vers la droite

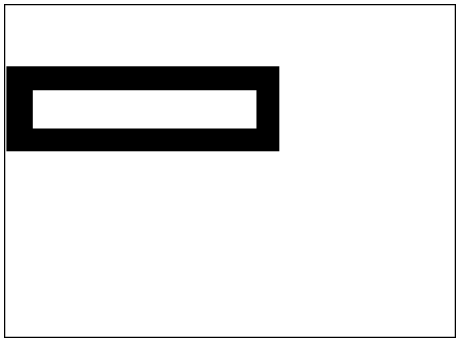


rectangle - 50 ` Déplacer le rectangle 50 pixels vers la gauche

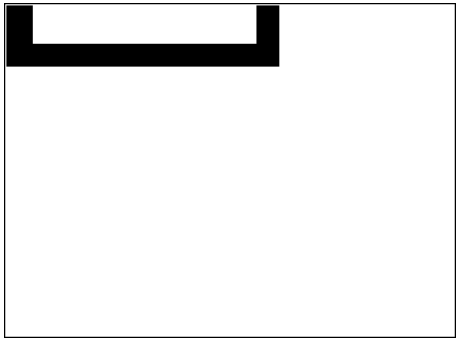


• Déplacement vertical

rectangle /50 ` Déplacer le rectangle 50 pixels vers le bas



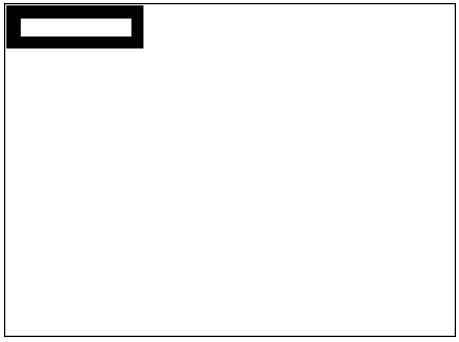
rectangle /-20 ` Déplacer le rectangle 20 pixels vers le haut



• Redimensionnement

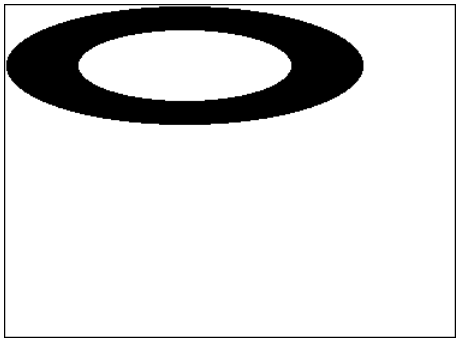
rectangle * 1.5 ` Augmenter la taille du rectangle de 50%

rectangle * 0.5 ` Réduire la taille du rectangle de 50%

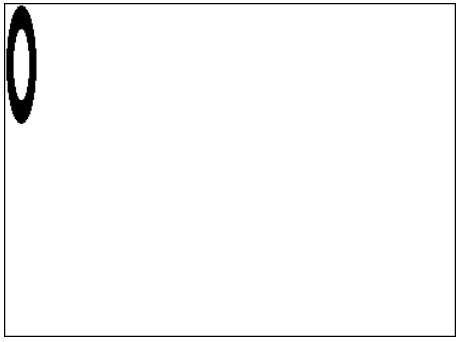


• Extension horizontale

cercle *+3 `Multiplier par 3 la largeur du cercle

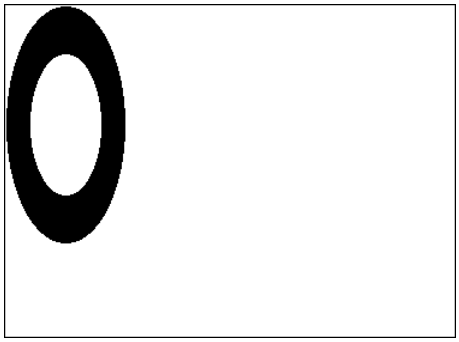


cercle *+ 0,25 ` La largeur du cercle est réduite à un quart de sa taille originale

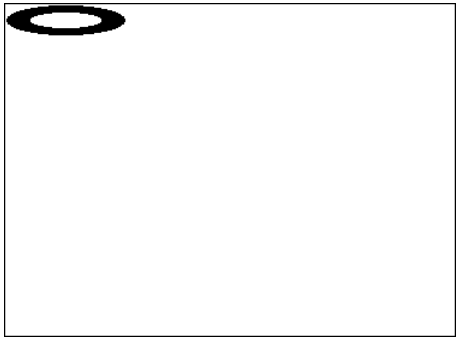


• Extension verticale

```
cercle */ 2 ` Doubler la hauteur du cercle
```



```
cercle */ 0.25 ` La hauteur du cercle est réduite à un quart de sa taille originale
```



Référence

Opérateurs, Opérateurs de comparaison, Opérateurs logiques, Opérateurs numériques, Opérateurs sur les bits, Opérateurs sur les chaînes, Opérateurs sur les dates, Opérateurs sur les heures.

Opérateurs sur les bits

Les opérateurs sur les bits s'appliquent à des expressions ou valeurs de type Entier long.

Note : Si vous passez une valeur de type Entier ou Numérique (Réel) à un opérateur sur les bits, 4e Dimension la convertit en Entier long avant de calculer le résultat de l'expression.

Lorsque vous employez des opérateurs sur les bits, vous devez considérer une valeur de type Entier long comme un tableau de 32 bits. Les bits sont numérotés de 0 à 31, de droite à gauche.

Comme un bit peut valoir 0 (zéro) ou 1, vous pouvez également considérer une valeur de type Entier long comme une expression dans laquelle vous pouvez stocker 32 valeurs de type Booléen. Lorsque le bit vaut 1, la valeur est Vrai et lorsque le bit vaut 0, la valeur est Faux.

Une expression utilisant un opérateur sur les bits retourne une valeur de type Entier long, à l'exception de l'opérateur Tester bit avec lequel l'expression retournée est du type Booléen. Le tableau suivant fournit la liste des opérateurs sur les bits et leur syntaxe :

Opération	Opérateur	Syntaxe	Retourne	
ET	&	E.long & E.long	E.long	
OU (inclusif)	|	E.long | E.long	E.long	
OU (exclusif)	^|	E.long ^| E.long	E.long	
Décaler bits à gauche	<<	E.long << E.long	E.long	(voir note 1)
Décaler bits à droite	>>	E.long >> E.long	E.long	(voir note 1)
Mettre bit à 1	?+	E.long ?+ E.long	E.long	(voir note 2)
Mettre bit à 0	?-	E.long ?- E.long	E.long	(voir note 2)
Tester bit	??	E.long ?? E.long	Booléen	(voir note 2)

Notes

(1) Dans les opérations utilisant Décaler bits à gauche et Décaler bits à droite, le second opérande indique le nombre de décalages de bits du premier opérande à effectuer dans la valeur retournée. Par conséquent, ce second opérande doit être compris entre 0 et 31. Notez qu'un décalage de 0 retourne une valeur inchangée et qu'un décalage de plus de 31 bits retourne 0x00000000 car tous les bits sont perdus. Si vous passez une autre valeur en tant que second opérande, le résultat sera non significatif.

(2) Dans les opérations utilisant Mettre bit à 1, Mettre bit à 0 et Tester bit, le second opérande indique le numéro du bit sur lequel agir. Par conséquent, ce second opérande doit être compris entre 0 et 31. Sinon, l'expression retourne inchangée la valeur du premier opérande pour Mettre bit à 1 et Mettre bit à 0, et retourne Faux pour Tester bit.

Le tableau suivant dresse la liste des opérateurs sur les bits et de leurs effets :

Opération sur les bits	Description
ET	Chaque bit retourné est le résultat de l'opération ET logique appliquée aux deux bits opérandes. Voici la table du ET logique : 1 & 1 → 1 0 & 1 → 0 1 & 0 → 0 0 & 0 → 0 En résumé, le résultat vaut 1 si les deux bits opérandes valent 1, dans tous les autres cas le bit résultant vaut 0.
OU (inclusif)	Chaque bit retourné est le résultat de l'opération OU inclusif logique appliquée aux deux bits opérandes. Voici la table du OU inclusif logique : 1 | 1 → 1 0 | 1 → 1 1 | 0 → 1 0 | 0 → 0 En résumé, le résultat vaut 1 si au moins un des deux bits opérandes vaut 1, sinon le bit résultant vaut 0.
OU (exclusif)	Chaque bit retourné est le résultat de l'opération OU exclusif logique appliquée aux deux bits opérandes. Voici la table du OU exclusif logique: 1 ^| 1 → 0 0 ^| 1 → 1 1 ^| 0 → 1 0 ^| 0 → 0 Donc, le résultat vaut 1 si un seul des deux bits opérandes vaut 1 (et pas l'autre), dans tous les autres cas le bit résultant vaut 0.
Décaler bits à gauche	La valeur retournée correspond au premier opérande dont la valeur est décalée vers la gauche du nombre de bits spécifié par le second opérande. Les bits auparavant situés à gauche sont perdus et les nouveaux bits situés à droite ont la valeur 0. **Note** : En ne tenant compte que des valeurs positives, un décalage vers la gauche de N bits revient à multiplier la valeur par 2^N.

Décaler bits à droite	La valeur retournée correspond au premier opérande dont la valeur est décalée vers la droite du nombre de bits spécifié par le second opérande. Les bits auparavant situés à droite sont perdus et les nouveaux bits situés à gauche ont la valeur 0. **Note** : En ne tenant compte que des valeurs positives, un décalage vers la droite de N bits revient à diviser la valeur par 2^N.
Mettre bit à 1	La valeur retournée est la valeur du premier opérande dans lequel le bit dont le numéro est spécifié par le second opérande est positionné à 1. Les autres bits sont inchangés.
Mettre bit à 0	La valeur retournée est la valeur du premier opérande dans lequel le bit dont le numéro est spécifié par le second opérande est positionné à 0. Les autres bits sont inchangés.
Tester bit	Retourne Vrai si, dans le premier opérande, le bit dont le numéro est indiqué par le second opérande vaut 1. Retourne Faux si, dans le premier opérande, le bit dont le numéro est indiqué par le second opérande vaut 0.

Exemples

(1) Le tableau ci-dessous propose un exemple d'utilisation de chaque opérateur sur les bits :

Opération	Exemple	Résultat
ET	0x0000FFFF & 0xFF00FF00	0x0000FF00
OU (inclusif)	0x0000FFFF | 0xFF00FF00	0xFF00FFFF
OU (exclusif)	0x0000FFFF ^| 0xFF00FF00	0xFF0000FF
Décaler bit gauche	0x0000FFFF << 8	0x00FFFF00
Décaler bit droit	0x0000FFFF >> 8	0x000000FF
Mettre bit à 1	0x00000000 ?+ 16	0x00010000
Mettre bit à 0	0x00010000 ?- 16	0x00000000
Tester bit	0x00010000 ?? 16	Vrai

(2) 4e Dimension exploite de nombreuses constantes prédéfinies. Le nom de certaines d'entre elles commence par "Bit" ou "Masque". C'est le cas des constantes incluses dans le thème Propriétés des ressources :

Constante	Type	Valeur
Masque ressource heap système	Entier long	64
Bit ressource heap système	Entier long	6
Masque ressource purgeable	Entier long	32
Bit ressource purgeable	Entier long	5
Masque ressource verrouillée	Entier long	16
Bit ressource verrouillée	Entier long	4
Masque ressource protégée	Entier long	8
Bit ressource protégée	Entier long	3
Masque ressource préchargée	Entier long	4
Bit ressource préchargée	Entier long	2
Masque ressource modifiée	Entier long	2
Bit ressource modifiée	Entier long	1

Ces constantes vous permettent de tester la valeur retournée par la fonction Lire proprietes ressource ou de définir la valeur à passer à ECRIRE PROPRIETES RESSOURCE. Les constantes dont le nom débute par “Bit” fournissent la position du bit que vous voulez tester, effacer ou fixer. Les constantes dont le nom débute par “Masque” sont des valeurs de type Entier long dans lesquelles seul le bit que vous voulez tester, effacer ou fixer est égal à 1.

Si, par exemple, vous devez tester si une ressource (dont les propriétés sont stockées dans la variable $vlResAttr) est purgeable ou non, vous pouvez écrire :

```
Si ($vlResAttr ?? Bit ressource purgeable) ` La ressource est-elle purgeable ?
```

ou :

```
Si (($vlResAttr & Masque ressource purgeable) # 0) ` La ressource est-elle purgeable ?
```

A l'inverse, vous pouvez utiliser ces constantes pour définir le même bit. Vous pouvez écrire :

```
$vlResAttr:=$vlResAttr ?+ Bit ressource purgeable
```

ou :

```
$vlResAttr:=$vlResAttr | Bit ressource purgeable
```

(3) Vous voulez stocker deux valeurs entières dans un Entier long. Vous pouvez écrire :

```
$vlLong:=($viIntA<<16) | $viIntB ` Stocker deux Entiers dans un Entier long
$vlIntA:=$vlLong>>16 ` Extraire l'Entier stocké dans le mot haut
$viIntB:=$vlLong & 0xFFFF ` Extraire l'entier stocké dans le mot bas
```

Note : Soyez prudent lorsque vous manipulez des Entiers longs ou des Entiers avec des expressions qui combinent des opérateurs sur les bits et des opérateurs numériques. Le bit supérieur (bit 31 pour un Entier long, bit 15 pour un Entier) détermine le signe de la valeur (positif s'il est à 0, négatif s'il est à 1). Les opérateurs numériques utilisent ce bit pour détecter le signe d'une valeur, mais les opérateurs sur les bits n'en tiennent pas compte.

Référence

Opérateurs, Opérateurs de comparaison, Opérateurs logiques, Opérateurs numériques, Opérateurs sur les chaînes, Opérateurs sur les dates, Opérateurs sur les heures, Opérateurs sur les images.

37

Pages formulaire

Les commandes de ce thème vous permettent de manipuler les pages des formulaires. 4D vous fournit des actions automatiques pour les boutons qui effectuent les mêmes tâches que les commandes PREMIERE PAGE, DERNIERE PAGE, PAGE SUIVANTE et PAGE PRECEDENTE. De plus, la version 6 propose une nouvelle action automatique équivalant à la commande ALLER A PAGE que vous pouvez associer aux objets tels que les onglets, les listes déroulantes, etc. A chaque fois que c'est possible, utilisez les actions automatiques pour les boutons plutôt que ces commandes.

Les commandes de gestion des pages peuvent être utilisées avec des formulaires entrée ou des formulaires affichés dans des boîtes de dialogue. Les formulaires sortie n'utilisent que la première page. Un formulaire comprend toujours au minimum une page, la première. Notez bien que quel que soit le nombre de pages qu'il contient, un formulaire ne peut être associé qu'à une seule méthode formulaire.

Vous pouvez utiliser la commande Page formulaire courante pour savoir quelle page est affichée à l'écran.

Note : Pendant que vous construisez un formulaire, vous pouvez utiliser les pages 1 à N du formulaire ainsi que la page zéro, dans laquelle vous placez les objets que vous voulez faire apparaître sur toutes les pages. Lors de l'utilisation du formulaire, et donc lorsque les commandes du thème Page formulaire sont appelées, seules les pages 1 à N sont accessibles : la page zéro est automatiquement combinée à la page affichée à l'écran.

PREMIERE PAGE

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

PREMIERE PAGE change la page courante d'un formulaire pour afficher la première page du formulaire. Si aucun formulaire n'est affiché, ou si la première page du formulaire est déjà affichée, PREMIERE PAGE ne fait rien.

Exemple

Cet exemple est une méthode appelée par une commande de menu. Elle affiche la première page du formulaire :

```
⇒ PREMIERE PAGE
```

Référence

ALLER A PAGE, DERNIERE PAGE, Page formulaire courante, PAGE PRECEDENTE, PAGE SUIVANTE.

DERNIERE PAGE

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

DERNIERE PAGE change la page courante d'un formulaire pour afficher la dernière page du formulaire. Si aucun formulaire n'est affiché ou si la dernière page du formulaire est déjà affichée, DERNIERE PAGE ne fait rien.

Exemple

Cet exemple est une méthode appelée par une commande de menu. Elle affiche la dernière page du formulaire courant :

```
⇒ DERNIERE PAGE
```

Référence

ALLER A PAGE, Page formulaire courante, PAGE PRECEDENTE, PAGE SUIVANTE, PREMIERE PAGE.

PAGE SUIVANTE

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

PAGE SUIVANTE change la page courante d'un formulaire pour afficher la page suivante. Si aucun formulaire n'est affiché, ou si la page affichée est la dernière page du formulaire, PAGE SUIVANTE ne fait rien.

Exemple

Cet exemple est une méthode appelée par une commande de menu. Elle provoque l'affichage la page qui suit celle qui actuellement affichée :

```
⇒ PAGE SUIVANTE
```

Référence

ALLER A PAGE, DERNIERE PAGE, Page formulaire courante, PAGE PRECEDENTE, PREMIERE PAGE.

PAGE PRECEDENTE

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

PAGE PRECEDENTE change la page courante d'un formulaire pour afficher la page précédente. Si aucun formulaire n'est affiché, ou si la page affichée est la première page du formulaire, PAGE PRECEDENTE ne fait rien.

Exemple

Cet exemple est une méthode appelée par une commande de menu. Elle provoque l'affichage la page qui précède celle qui est actuellement affichée :

```
⇒ PAGE PRECEDENTE
```

Référence

ALLER A PAGE, DERNIERE PAGE, Page formulaire courante, PAGE SUIVANTE, PREMIERE PAGE.

ALLER A PAGE (numéroPage)

Paramètre	Type		Description
numéroPage	Numérique	→	Numéro de la page à afficher

Description

ALLER A PAGE change la page courante d'un formulaire pour afficher la page désignée par numéroPage.

Si aucun formulaire n'est affiché, ALLER A PAGE ne fait rien. Si numéroPage est supérieur au nombre de pages du formulaire, la dernière page est affichée. Si numéroPage est inférieur à 1, la première page est affichée.

Exemple

L'exemple suivant est la méthode objet d'un bouton affichant la page 3 du formulaire :

```
⇒ ALLER A PAGE (3)
```

Référence

DERNIERE PAGE, Page formulaire courante, PAGE PRECEDENTE, PAGE SUIVANTE, PREMIERE PAGE.

Page formulaire courante → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Numéro de la page courante du formulaire actuellement affiché

Description

Page formulaire courante retourne le numéro de la page courante du formulaire actuellement affiché.

Exemple

Alors que vous être en train d'utiliser un formulaire, si vous choisissez une commande de menu ou si le formulaire reçoit un appel d'un autre process, vous voulez que des actions différentes soient effectuées en fonction de la page du formulaire affichée. Vous pouvez alors écrire :

```
   ` Méthode formulaire [maTable];"monFormulaire"
Au cas ou
   : (Evenement formulaire=Sur chargement)
      ` ...
   : (Evenement formulaire=Sur libération)
      ` ...
   : (Evenement formulaire=Sur menu sélectionné)
      $vlNuméroMenu:=Menu choisi >> 16
      $vlNuméroCmde:=Menu choisi & 0xFFFF
```

```
        Au cas ou
           : ($vlNuméroMenu=...)
              Au cas ou
                 : ($vlNuméroCmde=...)
⇒                   : (Page formulaire courante=1)
                       ` Effectuer une action appropriée pour la page 1
⇒                   : (Page formulaire courante=2)
                       ` Effectuer une action appropriée pour la page 2
                       ` ...
                 : ($vlNuméroCmde=...)
                       ` ...
              Fin de cas
           : ($vlNuméroMenu=...)
              ` ...
        Fin de cas
     : (Evenement formulaire=Sur appel extérieur)
        Au cas ou
⇒          : (Page formulaire courante=1)
              ` Fournir une réponse appropriée pour la page 1
⇒          : (Page formulaire courante=2)
              ` Fournir une réponse appropriée pour la page 2
        Fin de cas
           ` ...
  Fin de cas
```

Référence

ALLER A PAGE, DERNIERE PAGE, PAGE PRECEDENTE, PAGE SUIVANTE, PREMIERE PAGE.

38

Presse-Papiers

AJOUTER A PRESSE PAPIERS (typeDonnées; données)

Paramètre	Type		Description
typeDonnées	Alpha	→	Type de données (4 caractères)
données	BLOB	→	Données à ajouter au Presse-papiers

Description

AJOUTER A PRESSE PAPIERS ajoute dans le Presse-papiers les données du type spécifié dans typeDonnées présentes dans le BLOB données.

ATTENTION : La valeur passée dans typeDonnées établit la distinction majuscules/minuscules, par exemple “abcd” n'est pas égal à “ABCD.”

Si les données dans le BLOB sont correctement ajoutées au Presse-papiers, la variable OK prend la valeur 1. Sinon, la variable OK est mise à 0 et une erreur peut être générée.

Généralement, vous utilisez la commande AJOUTER A PRESSE PAPIERS pour placer plusieurs instances des mêmes données dans le Presse-papiers ou pour y ajouter des valeurs qui ne sont pas du type TEXT ou PICT. Pour ajouter de nouvelles données au Presse-papiers, il faut d'abord effacer le Presse-papiers à l'aide de la commande EFFACER PRESSE PAPIERS.

Si vous voulez effacer le Presse-papiers et y ajouter :
- du texte, utilisez la commande ECRIRE TEXTE DANS PRESSE PAPIERS,
- une image, utilisez la commande ECRIRE IMAGE DANS PRESSE PAPIERS.

Notez cependant que si un BLOB contient du texte ou une image, vous pouvez utiliser la commande AJOUTER A PRESSE PAPIERS pour y ajouter du texte ou une image.

Exemple

A l'aide des commandes du thème Presse-papiers et des BLOBs, vous pouvez écrire des méthodes de Couper/Copier/Coller pour gérer des données structurées au lieu d'une seule information. Dans l'exemple suivant, les deux méthodes projet écrire enregistrement dans Presse-papiers et lire enregistrement dans Presse-papiers vous permettent de traiter un enregistrement comme une information à copier dans le Presse-papiers.

```
` Méthode projet écrire enregistrement dans Presse-papiers
` écrire enregistrement dans Presse-papiers ( Numérique )
` écrire enregistrement dans Presse-papiers ( Numéro de table )

C_ENTIER LONG($1;$vlChamp;$vlTypeChamp)
C_POINTEUR($vpTable;$vpChamp)
C_ALPHA(255;$vaNomDoc)
C_TEXTE($vtDonnéesEnregistrement;$vtDonnéesChamp)
C_BLOB($vxDonnéesEnregistrement)

` Effacer le Presse-papiers (il restera vide s'il n'y a pas d'enregistrement courant)
EFFACER PRESSE PAPIERS
` Obtenir un pointeur vers la table dont le numéro est passé en paramètre
$vpTable:=Table($1)
` S'il y a un enregistrement courant pour cette table
Si ((Numero enregistrement($vpTable->)>=0) | (Numero enregistrement
                                                    ($vpTable->)=-3))
` Initialiser la variable Texte qui contiendra l'image de texte de l'enregistrement
$vtDonnéesEnregistrement:=""
   ` Pour chaque champ de l'enregistrement :
   Boucle ($vlChamp;1;Nombre de champs($1))
      ` Obtenir le type du champ
      LIRE PROPRIETES CHAMP($1;$vlChamp;$vlTypeChamp)
      ` Obtenir un pointeur vers le champ
      $vpChamp:=Champ($1;$vlChamp)
      ` Selon le type du champ, copier (ou non) ses données de façon appropriée
      Au cas ou
         : (($vlTypeChamp=Champ alphanumérique ) | ($vlTypeChamp=Texte ))
         $vtDonnéesChamp:=$vpChamp->
         : (($vlTypeChamp=Numérique ) | ($vlTypeChamp=Entier )
                                                | ($vlTypeChamp=Entier long)
                                          | ($vlTypeChamp=Champ ou variable date )
                                       | ($vlTypeChamp=Champ ou variable heure ))
            $vtDonnéesChamp:=Chaine($vpChamp->)
         : ($vlTypeChamp=Booléen )
            $vtDonnéesChamp:=Chaine(Num($vpChamp->);"Oui;;Non")
```

```
            Sinon
                  ` Passer et ignorer les autres types de champs
               $vtDonnéesChamp:=""
            Fin de cas
               ` Accumuler les données sur le champ dans une variable texte qui stocke
               ` l'image de texte de l'enregistrement
            $vtDonnéesEnregistrement:=$vtDonnéesEnregistrement+Nom du champ($1;
                                          $vlChamp)+":"+Caractere(9)+$vtDonnéesChamp+CR
               ` Note : La méthode CR retourne Caractere(13) sous MacOS et
               ` Caractere(13) + Caractere(10) sous Windows
         Fin de boucle
            ` Mettre l'image de texte de l'enregistrement dans le Presse-papiers
         ECRIRE TEXTE DANS PRESSE PAPIERS($vtDonnéesEnregistrement)
            ` Nommer le fichier d'Album dans le Dossier temporaire
         $vaNomDoc:=Dossier temporaire+"Album"+Chaine(1+(Hasard%99))
            ` Supprimer le fichier d'Album s'il existe (il faut tester une erreur ici)
         SUPPRIMER DOCUMENT($vaNomDoc)
            ` Créer le fichier d'Album
         REGLER SERIE(10;$vaNomDoc)
            ` Envoyer l'enregistrement entier dans le Presse-papiers
         ENVOYER ENREGISTREMENT($vpTable->)
            ` Fermer le fichier d'Album
         REGLER SERIE(11)
            ` Charger le fichier d'Album dans un BLOB
         DOCUMENT VERS BLOB($vaNomDoc;$vxDonnéesEnregistrement)
            ` Nous n'avons plus besoin du fichier d'Album
         SUPPRIMER DOCUMENT($vaNomDoc)
            ` Ajouter l'image complète de l'enregistrement dans le Presse-papiers
            ` Note: nous utilisons le type de données "4Drc" de façon arbitraire
⇒        AJOUTER A PRESSE PAPIERS("4Drc";$vxDonnéesEnregistrement)
            ` Le Presse-papiers contient :
            ` (1) Une image de texte de l'enregistrement
            ` (2) Une image entière de l'enregistrement (y compris les images, sous-tables et
            ` les champs de type BLOB)
      Fin de si
```

Lors de la saisie d'un enregistrement, si vous appliquez la méthode écrire enregistrement dans Presse-papiers à la table, le Presse-papiers contiendra le texte de l'enregistrement et également l'image entière de l'enregistrement.

Vous pouvez coller cette image de l'enregistrement dans un autre enregistrement, à l'aide de la méthode lire enregistrement dans Presse-papiers, qui est la suivante :

```
   ` Méthode lire enregistrement dans Presse-papiers
   ` lire enregistrement dans Presse-papiers ( Numéro )
   ` lire enregistrement dans Presse-papiers ( Numéro de table )
C_ENTIER LONG($1;$vlChamp;$vlTypeChamp;$vlPosCR;$vlPosColon)
C_POINTEUR($vpTable;$vpChamp)
C_ALPHA(255;$vaNomDoc)
C_BLOB($vxDonnéeesPressePapiers)
C_TEXTE($vtDonnéeesPressePapiers;$vtDonnéesChamp)

` Obtenir un pointeur vers la table dont le numéro est passé en tant que paramètre
$vpTable:=Table($1)
` S'il y a un enregistrement courant pour cette table
SI ((Numero enregistrement($vpTable->)>=0) | (Numero enregistrement
                                                    ($vpTable->)=-3))
   Au cas ou
         ` Est-ce que le Presse-papiers contient une image entière de l'enregistrement ?
      : (Tester presse papiers("4Drc")>0)
            ` Si oui, extraire le contenu du Presse-papiers
         LIRE PRESSE PAPIERS("4Drc";$vxDonnéesPressePapiers)
            ` Nommer le fichier d'Album dans le Dossier temporaire
         $vaNomDoc:=Dossier temporaire+"Album"+Chaine(1+(Hasard%99))
            ` Supprimer le fichier d'Album s'il existe (il faut tester l'erreur ici)
         SUPPRIMER DOCUMENT($vaNomDoc)
            ` Enregistrer le BLOB dans le fichier d'Album
         BLOB VERS DOCUMENT($vaNomDoc;$vxDonnéesPressePapiers)
            ` Ouvrir le fichier d'Album
         REGLER SERIE(10;$vaNomDoc)
            ` Recevoir l'enregistrement entier du fichier d'Album
         RECEVOIR ENREGISTREMENT($vpTable->)
            ` Fermer le fichier d'Album
         REGLER SERIE(11)
            ` Nous n'avons plus besoin du fichier d'Album
         SUPPRIMER DOCUMENT($vaNomDoc)
         ` Est-ce que le Presse-papiers contient du texte ?
      : (Tester presse papiers("TEXT")>0)
            ` Extraire le texte du Presse-papiers
         $vtDonnéesPressePapiers:=Lire texte dans presse papiers
               ` Initialiser le numéro de champ à incrémenter
         $vlChamp:=0
```

```
Repeter
   ` Chercher la ligne de champ suivante dans le texte
   $vlPosCR:=Position(CR ;$vtDonnéesPressePapiers)
   Si ($vlPosCR>0)
      ` Extraire la ligne de champ
      $vtDonnéesChamp:=Sous chaine($vtDonnéesPressePapiers;1;
                                                        $vlPosCR-1)
      ` S'il y a un signe deux points ":"
      $vlPosColon:=Position(":";$vtDonnéesChamp)
      Si ($vlPosColon>0)
         ` Ne récupérer que les données de champ (supprimer son nom)
         $vtDonnéesChamp:=Sous chaine($vtDonnéesChamp;
                                                        $vlPosColon+2)
      Fin de si
         ` Incrémenter le numéro du champ
      $vlChamp:=$vlChamp+1
         ` Le Presse-papiers peut contenir d'autres données dont nous n'avons
         ` pas besoin...
      Si ($vlChamp<=Nombre de champs($vpTable))
         ` Obtenir le type du champ
         LIRE PROPRIETES CHAMP($1;$vlChamp;$vlTypeChamp)
         ` Obtenir un pointeur vers le champ
         $vpChamp:=Champ($1;$vlChamp)
         ` Selon le type du champ, copier (ou non) le texte de manière
         ` appropriée
         Au cas ou
            : (($vlTypeChamp=Champ alphanumérique ) | ($vlTypeChamp=
                                                              Texte ))
               $vpChamp->:=$vtDonnéesChamp
            : (($vlTypeChamp=Numérique ) | ($vlTypeChamp=
                                          Entier ) | ($vlTypeChamp=Entier long ))
               $vpChamp->:=Num($vtDonnéesChamp)
            : ($vlTypeChamp=Champ ou variable date )
               $vpChamp->:=Date($vtDonnéesChamp)
            : ($vlTypeChamp=Champ ou variable heure )
               $vpChamp->:=Heure($vtDonnéesChamp)
            : ($vlTypeChamp=Booléen )
               $vpChamp->:=($vtDonnéesChamp="Oui")
         Sinon
               ` Passer et ignorer les autres types de données
         Fin de cas
```

```
            Sinon
               ` Tous les champs ont été affectés, sortir de la boucle
            $vtDonnéesPressePapiers:=""
            Fin de si
               ` Eliminer le texte qui vient d'être extrait
            $vtDonnéeesPressePapiers:=Sous chaine($vtDonnéeesPressePapiers;
                                                  $vlPosCR+Longueur(CR ))
         Sinon
            ` Aucun délimiteur trouvé, sortir de la boucle
         $vtDonnéesPressePapiers:=""
         Fin de si
         ` Répéter jusqu'à ce que nous avons des données
      Jusque (Longueur($vtDonnéesPressePapiers)=0)
   Sinon
      ALERTE("Le Presse-papiers ne contient pas de données pouvant être collées
                                                  en tant qu'enregistrement.")
   Fin de cas
Fin de si
```

Référence

ECRIRE IMAGE DANS PRESSE PAPIERS, ECRIRE TEXTE DANS PRESSE PAPIERS, EFFACER PRESSE PAPIERS.

Variables système

Si les données dans le BLOB sont correctement ajoutées au Presse-papiers, la variable système OK prend la valeur 1. Sinon, OK est mise à 0 et une erreur peut être générée.

Gestion des erreurs

S'il n'y a pas assez de mémoire pour ajouter les données du BLOB dans le Presse-papiers, une erreur -108 est générée.

EFFACER PRESSE PAPIERS

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

EFFACER PRESSE PAPIERS efface le contenu du Presse-papiers. Si le Presse-papiers contient plusieurs instances des mêmes données, toutes les instances sont effacées. Après un appel à EFFACER PRESSE PAPIERS, le Presse-papiers est vide.

Vous devez appeler EFFACER PRESSE PAPIERS une fois avant de placer des nouvelles données dans le Presse-papiers à l'aide de la commande AJOUTER A PRESSE PAPIERS, car cette dernière n'efface pas le Presse-papiers avant d'y coller des données.

Si vous appelez EFFACER PRESSE PAPIERS une fois et puis appelez AJOUTER A PRESSE PAPIERS plusieurs fois, vous pouvez couper ou copier les mêmes données sous des formats différents.

En revanche, les commandes ECRIRE TEXTE DANS PRESSE PAPIERS et ECRIRE IMAGE DANS PRESSE PAPIERS effacent automatiquement le Presse-papiers avant d'y placer des données TEXT ou PICT.

Exemple

(1) Le code suivant efface le Presse-papiers et puis y ajoute des données :

```
⇒ EFFACER PRESSE PAPIERS  ` Effacer le contenu du Presse-papiers
     ` Ajouter des données de type 'XWKZ'
  AJOUTER A PRESSE PAPIERS('XWKZ';$vxSomeData)
  AJOUTER A PRESSE PAPIERS('SYLK';$vxSylkData)  ` Ajouter des données de type SYLK
```

(2) Reportez-vous à l'exemple de la commande AJOUTER A PRESSE PAPIERS.

Référence

AJOUTER A PRESSE PAPIERS.

LIRE PRESSE PAPIERS (typeDonnées; données)

Paramètre	Type		Description
typeDonnées	Alpha	→	Type de données (4 caractères)
données	BLOB	←	Données extraites du Presse-papiers

Description

LIRE PRESSE PAPIERS retourne dans le champ ou la variable de type BLOB données les données qui se trouvent dans Presse-papiers, et dont le type est passé dans typeDonnées.

ATTENTION : La valeur passée dans typeDonnées établit la distinction majuscules/minuscules, par exemple “abcd” n'est pas égal à “ABCD.”

Si les données sont correctement extraites du Presse-papiers, la variable OK prend la valeur 1. Si le Presse-papiers est vide ou ne contient pas de données du type spécifié, la commande retourne un BLOB vide et la variable OK prend la valeur 0 ; l'erreur -102 est générée. S'il n'y a pas assez de mémoire pour extraire les données du Presse-papiers, la commande met la variable OK à 0 et génère l'erreur -108.

Exemple

Les méthodes objet suivantes sont celles de deux boutons qui copient et collent des données dans le tableau taOptions (pop up menu, liste déroulante...) se trouvant dans le formulaire :

```
    ` Méthode objet bCopiertaOptions
Si (Taille tableau(taOptions)>0)  ` Est-ce qu'il y a quelque chose à copier ?
        ` Mettre les éléments du tableau dans un BLOB
    VARIABLE VERS BLOB (taOptions;$vxClipData)
    EFFACER PRESSE PAPIERS  ` Vider le Presse-papiers
        ` Le type de données est choisi arbitrairement
    AJOUTER A PRESSE PAPIERS ("artx";taOptions)
Fin de si
```

```
      ` Méthode objet bCollertaOptions
      ` Est-ce qu'il y a les données du type "artx" dans le Presse-papiers ?
   Si (Tester presse papiers ("artx")>0)
⇒     LIRE PRESSE PAPIERS ("artx";$vxClipData)  ` Extraire les données du Presse-papiers
         ` Remplir le tableau avec les données venant du BLOB
      BLOB VERS VARIABLE ($vxClipData;taOptions)
      taOptions:=0  ` Réinitialiser l'élément sélectionné du tableau
   Fin de si
```

Référence

AJOUTER A PRESSE PAPIERS, LIRE IMAGE DANS PRESSE PAPIERS, Lire texte dans presse papiers.

Variables système

Si les données sont extraites correctement, la variable OK prend la valeur 1. Sinon, elle prend la valeur 0 et une erreur est générée.

Gestion des erreurs

- S'il n'y a pas assez de mémoire pour extraire les données, l'erreur -108 est générée.
- S'il n'y a pas de données du type spécifié dans le Presse-papiers, l'erreur -102 est générée.

LIRE IMAGE DANS PRESSE PAPIERS (image)

Paramètre	Type		Description
image	Image	←	Image extraite du Presse-papiers

Description

LIRE IMAGE DANS PRESSE PAPIERS retourne l'image présente dans le Presse-papiers dans le champ ou la variable image.

Si l'image est correctement extraite du Presse-papiers, la variable OK prend la valeur 1. Si le Presse-papiers est vide ou ne contient pas d'image, la variable OK prend la valeur 0 et l'erreur –102 est générée. S'il n'y a pas assez de mémoire, la commande donne la valeur 0 à la variable OK et génère l'erreur –108.

Exemple

Ci-dessous, la méthode objet d'un bouton affecte l'image présente dans le Presse-papiers, s'il y en a une, au champ [Employés]Photo :

```
    Si (Tester presse papiers ("PICT")>0)
⇒       LIRE IMAGE DANS PRESSE PAPIERS ([Employés]Photo)
    Sinon
       ALERTE ("Le Presse-papiers ne contient pas d'image.")
    Fin de si
```

Référence

LIRE PRESSE PAPIERS, Lire texte dans presse papiers, Tester presse papiers.

Variables et ensembles système

Si l'image est correctement extraite, OK prend la valeur 1. Sinon, OK prend la valeur 0 et une erreur est générée.

Gestion des erreurs

- S'il n'y a pas assez de mémoire pour extraire l'image, l'erreur -108 est générée.
- Si aucune photo ne se trouve dans le Presse-papiers, l'erreur -102 est générée.

Lire texte dans presse papiers

Lire texte dans presse papiers → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Texte présent dans le Presse-papiers

Description

Lire texte dans presse papiers retourne le texte présent dans le Presse-papiers.

Si le texte est correctement extrait du Presse-papiers, la variable OK prend la valeur 1. Si le Presse-papiers est vide ou ne contient pas de texte, la commande retourne une chaîne vide, la variable OK prend la valeur 0 et l'erreur –102 est générée. S'il n'y a pas assez de mémoire, la variable OK prend la valeur 0 et l'erreur –108 est générée.

Les champs et variables de type Texte de 4e Dimension peuvent contenir jusqu'à 32 000 caractères. Si le Presse-papiers en contient davantage, le résultat retourné par Lire texte dans presse papiers est tronqué au moment d'être collé dans le champ ou variable. Pour gérer des textes longs venant du Presse-papiers, testez en premier lieu la taille des données à l'aide de la fonction Tester presse papiers. Ensuite, si le texte dépasse 32 000 caractères, utilisez la commande LIRE PRESSE PAPIERS au lieu de Lire texte dans presse papiers.

Exemple

L'exemple teste s'il y a du texte dans le Presse-papiers. En fonction de la taille des données, la méthode extrait le texte du Presse-papiers soit comme du texte, soit comme un BLOB :

```
$vlTaille:=Tester presse papiers ("TEXT")
Au cas ou
   : ($vlTaille<=0)
      ALERTE ("Il n'y a pas de texte dans le presse-papiers.")
   : ($vlTaille<=32000)
⇒     $vlDonnéesPresse:=Lire texte dans presse papiers
      Si (OK=1)
         ` Traitement du texte
      Fin de si
   : ($vlTaille>32000)
      LIRE PRESSE PAPIERS ("TEXT";$vxDonnéesPresse)
      Si (OK=1)
         ` Traitement du BLOB
      Fin de si
Fin de cas
```

Référence

LIRE IMAGE DANS PRESSE PAPIERS, LIRE PRESSE PAPIERS, Tester presse papiers.

Variables et ensembles système

Si le texte est correctement extrait, la variable système OK prend la valeur 1, sinon elle prend la valeur 0 et une erreur est générée.

Gestion des erreurs

- S'il n'y a pas assez de mémoire pour extraire le texte, l'erreur –108 est générée.
- Si aucun texte ne se trouve dans le Presse-papiers, l'erreur –102 est générée.

ECRIRE IMAGE DANS PRESSE PAPIERS (image)

Paramètre	Type		Description
image	Image	→	Image à copier dans le Presse-papiers

Description

ECRIRE IMAGE DANS PRESSE PAPIERS place une copie de l'image que vous avez passée dans image dans le Presse-papiers. Les données éventuellement présentes dans le Presse-papiers sont préalablement effacées.

Après avoir placé l'image dans le Presse-papiers, vous pouvez la récupérer à l'aide de la commande LIRE IMAGE DANS PRESSE PAPIERS ou LIRE PRESSE PAPIERS("PICT";...).

Si l'image est correctement collée dans le Presse-papiers, la variable système OK prend la valeur 1. S'il n'y a pas assez de mémoire pour coller l'image dans le Presse-papiers, la variable système OK prend la valeur 0, mais aucune erreur n'est générée.

Exemple

Dans une fenêtre flottante, vous affichez un formulaire contenant le tableau tabNomEmployés qui liste les noms des employés stockés dans la table [Employés]. Chaque fois que vous cliquez sur un nom, vous voulez copier la photographie de l'employé dans le Presse-papiers. Dans la méthode objet du tableau, vous écrivez :

```
Si (tabNomEmployés#0)
   CHERCHER ([Employés];[Employés]Nom=tabNomEmployés{tabNomEmployés})
   Si (Taille image ([Employés]Photo)>0)
⇒     ECRIRE IMAGE DANS PRESSE PAPIERS ([Employés]Photo) ` Copier la photo
   Sinon
      EFFACER PRESSE PAPIERS  ` Aucune photo ou aucun enregistrement trouvé(e)
   Fin de si
Fin de si
```

Référence

AJOUTER A PRESSE PAPIERS, LIRE IMAGE DANS PRESSE PAPIERS.

Variables et ensembles système

Si une copie de l'image est correctement collée dans le Presse-papiers, la variable système OK prend la valeur 1.

ECRIRE TEXTE DANS PRESSE PAPIERS

ECRIRE TEXTE DANS PRESSE PAPIERS (texte)

Paramètre	Type		Description
texte	Alpha	→	Texte à coller dans le Presse-papiers

Description

ECRIRE TEXTE DANS PRESSE PAPIERS place une copie du texte que vous avez passé dans texte dans le Presse-papiers. Les données éventuellement présentes dans le Presse-papiers sont auparavant effacées.

Vous pouvez récupérer le texte collé dans le Presse-papiers à l'aide de la fonction Lire texte dans presse papiers ou en appelant LIRE PRESSE PAPIERS ("TEXT";...).

Si le texte est correctement collé dans le Presse-papiers, la variable système OK prend la valeur 1. S'il n'y a pas assez de mémoire pour placer une copie du texte dans le Presse-papiers, la variable système OK prend la valeur 0, mais aucune erreur n'est générée.

Les expressions de type Texte de 4e Dimension peuvent contenir jusqu'à 32 000 caractères. Pour copier des valeurs de texte plus importantes, il faut placer le texte dans un BLOB, appeler EFFACER PRESSE PAPIERS puis AJOUTER A PRESSE PAPIERS ("TEXT";...).

Exemple

Référez-vous à l'exemple de la commande AJOUTER A PRESSE PAPIERS.

Référence

AJOUTER A PRESSE PAPIERS, Lire texte dans presse papiers.

Variables et ensembles système

Si la copie du texte est correctement collée dans le Presse-papiers, la variable système OK prend la valeur 1.

Tester presse papiers

version 6.0

Tester presse papiers (typeDonnées) → Numérique

Paramètre	Type		Description
typeDonnées	Alpha	→	Type de données (4 caractères)
Résultat	Numérique	←	Taille (en octets) des données contenues dans le Presse-papiers ou code d'erreur

Description

Tester presse papiers vous permet de savoir s'il y a des données du type typeDonnées dans le Presse-papiers.

ATTENTION : La valeur passée dans typeDonnées établit la distinction majuscules/minuscules, par exemple “abcd” n'est pas égal à “ABCD.”

Si le Presse-papiers est vide ou ne contient pas de données du type spécifié, la fonction retourne une erreur -102 (référez-vous ci-dessous à la table des constantes). Si le Presse-papiers contient des données du type spécifié, la fonction retourne la taille des données exprimée en octets.

Après avoir vérifié que le Presse-papiers contient bien des données du type que vous voulez, vous pouvez les récupérer à l'aide d'une des commandes suivantes :

• Si le Presse-papiers contient du texte, vous pouvez l'extraire à l'aide de la commande Lire texte dans presse papiers, qui retourne une valeur texte. Sinon, vous pouvez utiliser la commande LIRE PRESSE PAPIERS, qui retourne le texte dans un BLOB.

• Si le Presse-papiers contient une image, vous pouvez l'extraire à l'aide de la commande LIRE IMAGE DANS PRESSE PAPIERS, qui retourne l'image dans un champ ou une variable. Sinon, vous pouvez utiliser la commande LIRE PRESSE PAPIERS, qui retourne l'image dans un BLOB.

• Pour tout type de données, vous pouvez utiliser la commande LIRE PRESSE PAPIERS, qui retourne les données dans un BLOB.

4e Dimension fournit les constantes suivantes :

Constante	Type	Valeur
Données absentes presse-papiers	Entier long	-102
Données texte	Alpha	TEXT
Données image	Alpha	PICT

Exemples

(1) L'exemple suivant teste si le Presse-papiers contient une image et, si oui, la copie dans une variable 4D :

```
⇒      Si (Tester presse papiers(Données image)>0) `Y a-t-il une image dans le presse-papiers ?
          LIRE IMAGE DANS PRESSE PAPIERS ($vPicVariable) ` Si oui, en extraire l'image
       Sinon
          ALERTE("Il n'y a pas d'image dans le Presse-papiers.")
       Fin de si
```

(2) Généralement, après un couper ou un copier, les applications placent des données de type Texte ou Image dans le Presse-papiers, ces deux types de données standard sont reconnus par la plupart des applications. Cependant, une application peut placer dans le Presse-papiers plusieurs copies des mêmes données sous des formats différents. Par exemple, chaque fois que vous copiez ou coupez un tableau, l'application tableur peut placer les données dans un format propriétaire — par exemple, 'SPSH' — ou dans les formats SYLK et TEXT. La copie 'SPSH' contient les données structurées dans le format interne de l'application. La copie SYLK contient les mêmes données, mais dans le format SYLK, reconnu par la plupart des tableurs. Enfin, la copie TEXT contient les mêmes données, mais sans les informations de formatage supplémentaires présentes dans les formats SYLK ou 'SPSH'. Donc, lorsque vous écrivez des routines de Couper/Copier/Coller entre 4e Dimension et une application tableur, en prenant l'hypothèse que vous connaissez la description du format 'SPSH' et que vous pouvez analyser les données SYLK, vous pouvez écrire le code suivant :

```
       Au cas ou
             ` D'abord, vérifier si le Presse-papiers contient les données venant du tableur
⇒          : (Tester presse papiers ('SPSH') > 0)
             `...
             ` Ensuite, vérifier si le Presse-papiers contient des données au format SYLK
⇒          : (Tester presse papiers ('SYLK') > 0)
             `...
             ` Enfin, vérifier si le Presse-papiers contient des données au format TEXT
⇒          : (Tester presse papiers ('TEXT') > 0)
             `...
       Fin de cas
```

Autrement dit, vous essayez d'extraire du Presse-papiers la copie des données la plus riche en informations originales.

(3) Référez-vous à l'exemple de la commande AJOUTER A PRESSE PAPIERS.

Référence

LIRE IMAGE DANS PRESSE PAPIERS, LIRE PRESSE PAPIERS, Lire texte dans presse papiers.

39

Process

Introduction aux process

Le multi-tâche dans 4D représente la possibilité d'exécuter simultanément plusieurs opérations de base de données distinctes. Ces opérations sont appelées des **process**.

Créer de multiples process équivaut à avoir plusieurs utilisateurs travaillant sur le même ordinateur, chacun effectuant sa tâche. Cela signifie principalement qu'une méthode peut être exécutée comme une tâche distincte de base de données.

Cette section traite les sujets suivants:
• Créer et effacer des process
• Eléments du process
• Process créés par 4D
• Process locaux et globaux
• Verrouillage d'enregistrements entre process.

Note : Cette section ne traite pas les procédures stockées. Pour cela, reportez-vous à la section Procédures stockées dans le manuel de référence de 4D Server.

Créer un process séparé

Il existe trois manières de créer un nouveau process :
• Exécuter une méthode à partir du mode Utilisation en sélectionnant la case à cocher **Nouveau Process** dans la boîte de dialogue d'exécution de méthode. La méthode choisie dans ce dialogue est la méthode process.
• Les process peuvent être démarrés par les commandes de menu. Dans l'éditeur de menus de l'environnement Structure, sélectionnez la commande de menu et cochez l'option **Démarrer un process**. La méthode associée à la commande de menu est la méthode process.
• Utiliser la fonction Nouveau process. La méthode passée en paramètre à la fonction Nouveau process est la méthode process.

Un process peut être supprimé dans les conditions suivantes. Les deux premières sont automatiques :
• Lorsque la méthode process a terminé de s'exécuter,
• Lorsque l'utilisateur quitte la base,
• Si vous stoppez le process par le langage ou utilisez le bouton **Stop** dans le débogueur,
• Si vous choisissez **Tuer** dans le menu **Process** du mode Structure.

Un process peut créer un autre process. Les process ne sont pas organisés hiérarchiquement. Tous les process sont égaux, et cela indépendamment du process à partir duquel ils ont été créés. Une fois créé, le process vit indépendamment de celui qui l'a créé.

Eléments d'un process

Chaque process contient certains éléments spécifiques. Il y a trois types d'éléments bien distincts dans un process :
• **Eléments d'interface** : Ce sont les éléments nécessaires à l'affichage du process.
• **Eléments de données** : Ce sont les informations liées aux données de la base.
• **Eléments de langage** : Ce sont les éléments utilisés par le langage ou importants pour le développement de l'application.

Eléments d'interface

Les éléments d'interface sont les suivants :
• **Barre de menus** : Chaque process a sa propre barre de menus courante. La barre de menus du process de premier plan est la barre de menu courante de la base.
• **Une ou plusieurs fenêtres** : Chaque process peut contrôler plusieurs fenêtres ouvertes simultanément. A l'inverse, des process peuvent n'avoir pas de fenêtre du tout.
• **Une fenêtre de premier plan** : Bien qu'un process puisse disposer de plusieurs fenêtres ouvertes simultanément, chaque process n'a qu'une fenêtre active. Pour avoir plusieurs fenêtres actives à la fois, vous devez démarrer plusieurs process.

Eléments de données

Les éléments de données se réfèrent aux données de la base. Ce sont les suivants :
• **Sélection courante par table** : Chaque process a sa propre sélection courante. La même table peut avoir differentes sélections courantes dans différents process.
• **Enregistrement courant par table** : Chaque table peut avoir un enregistrement courant différent dans chaque process.

Note : Cette description des éléments de données est valide si les process sont des process globaux. Par défaut, tous les process sont globaux. Reportez-vous plus bas au paragraphe traitant de ce point.

Eléments de langage

Les éléments de langage d'un process sont tous les éléments liés à la programmation dans 4D.
• **Variables** : Chaque process a ses propres variables process. Reportez-vous à la section Variables pour plus d'informations. Les variables process ne sont reconnues que dans le cadre de leur process natif.
• **Table par défaut** : Chaque process a sa propre table par défaut. Cependant, notez que la commande TABLE PAR DEFAUT est seulement une convention de programmation.

• **Formulaires entrée et sortie** : Les formulaires entrée et sortie par défaut peuvent être choisis par programmation pour chaque table et chaque process.
• **UserSet, LockedSet et ensembles process** : Chaque process a ses propres ensembles process. UserSet et LockedSet sont des ensembles process. Les ensembles process sont effacés dès que la méthode du process est terminée.
• **Appel sur erreur par process** : Chaque process a sa propre méthode de gestion d'erreurs.
• **Fenêtre de débogueur** : Chaque process peut avoir sa propre Fenêtre de débogueur.

Process utilisateur

Les process utilisateur sont des process que vous créez pour effectuer certaines tâches. Ils partagent le temps machine avec les process principaux. Les process de connexion Web sont des process utilisateur.

Voici les process créés et gérés par 4D :
• **Process principal** : Le process principal gère les environnements Utilisation et Menus Créés. L'écran d'accueil par défaut de l'environnement Menus créés est aussi une partie du process principal. Ce process est créé à l'ouverture d'une session 4D.
• **Process de structure** : Le process de structure gère le mode Structure, qui fonctionne dans une process séparé. Le process de structure peut être fermé par la commande **Quitter le mode Structure** du menu **Fichier** en mode Structure. Il n'y a pas de process de structure dans une base compilée. Le process de structure est créé uniquement lorsque l'utilisateur entre dans l'environnement structure pour la première fois de la session. Si l'application s'ouvre par défaut en mode Utilisation ou Menus créés, le process n'est pas créé.
• **Process Server Web** : Le process Server Web est créé lorsque la base est publiée sur le Web. Reportez-vous à la section Services Web, Process de connexion Web pour plus d'informations.
• **Process Gestionnaire du cache** : Le process Gestionnaire du cache gère les entrées/sorties disque de la base. Ce process est créé dès que 4D ou 4D Server est lancé.
• **Process d'indexation** : Le process d'indexation gère les index des champs de la base dans un process séparé. Ce process est créé lorsqu'un index est créé ou détruit pour un champ.
• **Process de Gestion d'événements** : Ce process est créé quand une méthode de gestion d'événement est installée par la commande APPELER SUR EVENEMENT. Ce process exécute la méthode d'appel sur événement installée par la commande APPELER SUR EVENEMENT à chaque fois qu'un événement se produit. La méthode événement est la méthode process de ce process. Elle s'exécute continuellement, même s'il n'y a pas de méthode en exécution. La gestion d'événements fonctionne aussi en mode Structure.

Process globaux et locaux

La portée (l'aire d'action) des process peut être globale ou locale. Par défaut, tous les process sont globaux.

Dans la plupart des cas, vous utiliserez des process globaux. Les process globaux peuvent effectuer n'importe quelle opération, accéder aux données et les manipuler.

Les process locaux ne doivent être utilisés que pour des opérations qui n'accèdent pas aux données. Par exemple, vous pouvez utiliser un process local pour contrôler les éléments d'interface comme les palettes flottantes ou exécuter une méthode de gestion d'événements.

Attention : Si vous tentez d'accéder aux données à partir d'un process local, vous accédez aux données par l'intermédiaire du Process principal, et prenez donc le risque d'entrer en conflit avec les opérations effectuées dans ce process.

Vous spécifiez qu'un process est local par le biais de son nom. Le nom d'un process local doit commencer par le symbole dollar ($).

4D Server : Avec 4D Server, l'utilisation de process locaux côté client, pour des opérations qui ne nécessitent pas d'accès aux données, permet d'allouer davantage de temps machine à des tâches qui sollicitent intensivement le serveur.

Verrouillage d'enregistrements entre process

Un enregistrement est verrouillé pour un process lorsqu'un autre process l'a déjà chargé pour modification. Un enregistrement verrouillé peut être chargé par un autre process mais ne peut pas être modifié. L'enregistrement est déverrouillé seulement dans le process dans lequel l'enregistrement est modifié. Une table doit être en mode lecture/écriture pour qu'un enregistrement puisse être chargé non verrouillé.

Référence

Méthodes, Méthodes projet, Variables.

Nouveau process

Process

version 6.0 (Modifiée)

Nouveau process (méthode; pile{; nom{; param{; param2; ...; paramN}{; *}}}) → Numérique

Paramètre	Type		Description
méthode	Alpha	→	Méthode à exécuter dans le process
pile	Numérique	→	Taille de la pile en octets
nom	Alpha	→	Nom du process créé
param	Expression	→	Paramètre(s) de la méthode
*		→	Process unique
Résultat	Numérique	←	Numéro du process nouvellement créé ou du process déjà en cours d'exécution

Description

La commande Nouveau process lance un nouveau process (sur la même machine) et retourne le numéro de ce process.

Si le process n'a pas pu être créé, par exemple s'il n'y a pas assez de mémoire, Nouveau process retourne zéro et une erreur est générée. Vous pouvez intercepter cette erreur à l'aide d'une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Méthode du process : Vous passez le nom de la méthode de gestion du nouveau process dans méthode. Une fois que 4D a défini le contexte pour le nouveau process, il démarre l'exécution de cette méthode qui devient alors la méthode du process.

Pile du process : Vous passez dans pile la quantité de mémoire allouée pour la pile du process. Cette valeur représente la place utilisée en mémoire pour “empiler” les appels de méthode, les variables locales, les paramètres des sous-routines et les enregistrements empilés. Elle est exprimée en octets, vous devez généralement passer au moins 32K (environ 32000 octets) mais vous pouvez passer davantage si la chaîne d'appel dans le process (sous-routines appelant des sous-routines en cascade) est importante. Si nécessaire, vous pouvez passer par exemple 200K (environ 200000 bytes).

Note : La pile n'est pas la mémoire totale réservée au process. Les process se partagent la mémoire pour les enregistrements, les variables interprocess, etc. Un process utilise également de la mémoire supplémentaire pour stocker ses variables process. La pile contient diverses informations internes à 4D ; la taille de ces informations varie en fonction du nombre d'appels de méthodes imbriquées.

Nom du process : Vous passez le nom du nouveau process dans nom. Ce nom s'affichera dans la fenêtre **Liste des process** du mode Structure de 4D et sera retourné par la commande INFORMATIONS PROCESS. Un nom de process peut contenir jusqu'à 31 caractères. Vous pouvez omettre ce paramètre ; dans ce cas, le nom du process sera une chaîne vide. Vous pouvez créer un process local en préfixant son nom d'un symbole dollar ($).

Important : Rappelez-vous que, en client/serveur, les process locaux ne doivent pas accéder aux données.

Paramètres de la méthode process : Depuis la version 6 de 4D, vous pouvez passer des paramètres à la méthode process. Vous pouvez le faire de la même manière que pour les sous-routines. Notez cependant qu'il y a une restriction : vous ne pouvez pas passer d'expression de type Pointeur. Rappelez-vous également que les tableaux ne peuvent pas être passés comme paramètres à une méthode. Une fois qu'elle a commencé à s'exécuter dans le contexte du nouveau process, la méthode process reçoit les valeurs des paramètres dans $1, $2, etc.

Note : Si vous passez des paramètres à la méthode process, vous devez passer le paramètre nom, il ne peut être omis dans ce cas.

Si vous passez le dernier paramètre (optionnel) *, vous indiquez à 4D de vérifier en premier lieu si un process du même nom que celui que vous avez passé dans nom est déjà en cours d'exécution. Si c'est le cas, 4D ne démarre pas de nouveau process et retourne le numéro du process existant.

Exemple

Examinons la méthode projet suivante :

```
   ` AJOUT CLIENTS
CHANGER BARRE (1)
Repeter
   AJOUTER ENREGISTREMENT([Clients];*)
Jusque (OK=0)
```

Si vous associez cette méthode projet à une commande de menu créé dans l'Editeur de barres de menus en mode Structure et que vous lui affectez la propriété **Démarrer un process**, 4D va automatiquement créer un nouveau process lors de l'exécution de la méthode. L'instruction CHANGER BARRE(1) associe cette barre de menus au nouveau process. En l'absence de toute fenêtre (que vous pourriez avoir ouverte avec Creer fenetre), l'appel à AJOUTER ENREGISTREMENT en créera une automatiquement.

Si maintenant vous voulez pouvoir démarrer le process Ajout Clients lorsque vous cliquez sur un bouton situé dans un tableau de contrôle personnalisé, vous pouvez écrire :

```
      ` Méthode objet bouton bAjoutClients
⇒     $vlProcessID:=Nouveau process("Ajout Clients";32*1024;"Ajout de clients")
```

Ce bouton fait la même chose que la commande de menu personnalisée.

Si, maintenant, lorsque la commande de menu est sélectionnée ou lorsque le bouton reçoit un clic, vous voulez que le process soit lancé s'il n'existe pas ou qu'il soit passé au premier plan s'il existe déjà, vous pouvez créer la méthode DEMARRER AJOUT CLIENTS :

```
      ` DEMARRER AJOUT CLIENTS
⇒     $vlProcessID:=Nouveau process("Ajout Clients";32*1024;"Ajout de clients ";*)
      Si ($vlProcessID#0)
         PASSER AU PREMIER PLAN ($vlProcessID)
      Fin de si
```

La méthode objet de bAjoutClient devient :

```
      ` Méthode objet bouton bAjoutClients
   DEMARRER AJOUT CLIENTS
```

Dans l'éditeur de barres de menus, vous remplacez AJOUT CLIENTS par la méthode DEMARRER AJOUT CLIENTS. Désélectionnez l'option **Démarrer un process** pour la commande de menu.

Référence

Executer sur serveur, Introduction aux process, Méthodes, Méthodes projet, Variables.

Executer sur serveur

Process

version 6.0

Executer sur serveur (procédure; pile{; nom{; param{; param2; ...; paramN}{; *}}}) → Numérique

Paramètre	Type		Description
procédure	Alpha	→	Procédure à exécuter dans le process
pile	Numérique	→	Taille de la pile en octets
nom	Alpha	→	Nom du process créé
param	Expression	→	Paramètre(s) de la procédure
*		→	Process unique
Résultat	Numérique	←	Numéro du process pour un process nouvellement créé ou un process déjà en cours d'exécution

Description

La commande Executer sur serveur lance un nouveau process sur la machine serveur (lorsqu'elle est appelée en environnement client/serveur) et retourne le numéro de ce process.

Executer sur serveur vous permet de démarrer une procédure stockée. Pour plus d'informations sur les procédures stockées, reportez-vous à la section Procédures stockées dans le manuel de référence de 4D Server.

Si vous appelez Executer sur serveur sur un poste client, la commande retourne un numéro de process négatif. Si vous appelez Executer sur serveur sur le poste serveur, la commande retourne un numéro de process positif. A noter que l'appel de la fonction Nouveau process sur le poste serveur est équivalent à l'appel de Executer sur serveur.

Si le process n'a pas pu être créé (par exemple s'il n'y a pas assez de mémoire), Executer sur serveur retourne zéro et une erreur est générée. Vous pouvez intercepter cette erreur à l'aide d'une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Méthode du process : Vous passez le nom de la méthode de gestion du nouveau process dans méthode. Une fois que 4D a défini le contexte pour le nouveau process, il démarre l'exécution de cette méthode qui devient alors la méthode du process.

Pile du process : Vous passez dans pile la quantité de mémoire allouée pour la pile du process. Cette valeur représente la place utilisée en mémoire pour “empiler” les appels de méthodes, les variables locales, les paramètres des sous-routines et les enregistrements empilés. Elle est exprimée en octets, vous devez généralement passer au moins 32K (environ 32000 octets) mais vous pouvez passer davantage si la chaîne d'appel dans le process (sous-routines appelant des sous-routines en cascade) est importante. Si nécessaire, vous pouvez passer par exemple 200K (environ 200000 octets).

Note : La pile n'est pas la mémoire totale réservée au process. Les process se partagent la mémoire pour les enregistrements, les variables interprocess, etc. Un process utilise également de la mémoire supplémentaire pour stocker ses variables process. La pile ne contient que les variables locales, les appels de méthodes, les paramètres des sous-routines et les enregistrements empilés.

Nom du process : Vous passez le nom du nouveau process dans nom. Avec 4D monoposte, ce nom s'affichera dans la fenêtre **Liste des process** du mode Structure de 4D et sera retourné par la commande INFORMATIONS PROCESS appliquée à ce process. En client/serveur, ce nom apparaîtra en bleu dans la liste des **Procédures stockées** de la fenêtre principale de 4D Server.

Un nom de process peut contenir jusqu'à 31 caractères. Vous pouvez omettre ce paramètre ; dans ce cas, le nom du process sera une chaîne vide.

Attention : A la différence de la commande Nouveau process, vous ne devez pas avec Executer sur serveur créer un process local en préfixant son nom du symbole dollar ($). Cela fonctionnerait correctement en version monoposte, car Executer sur serveur se comporte comme Nouveau process dans cet environnement, mais, en client/serveur, cela génèrerait une erreur.

Paramètres de la méthode process : Depuis la version 6 de 4D, vous pouvez passer des paramètres à la méthode process. Vous pouvez le faire de la même manière que pour les sous-routines. Notez cependant qu'il y a une restriction : vous ne pouvez pas passer d'expression de type Pointeur. Rappelez-vous également que les tableaux ne peuvent pas être passés comme paramètres à une méthode. Une fois qu'elle a commencé à s'exécuter dans le contexte du nouveau process, la méthode process reçoit les valeurs des paramètres dans $1, $2, etc.

Note : Si vous passez des paramètres à la méthode process, vous devez passer le paramètre nom, il ne peut être omis dans ce cas.

Paramètre optionnel * : Si vous passez le dernier paramètre (optionnel) *, vous indiquez à 4D de vérifier en premier lieu si un process du même nom que celui que vous avez passé dans nom est déjà en cours d'exécution. Si c'est le cas, 4D ne démarre pas de nouveau process et retourne le numéro du process existant.

Exemple

L'exemple suivant démontre comment l'import de données peut être accéléré de manière spectaculaire en environnement client/serveur. La méthode Import classique listée ci-dessous vous permet de mesurer combien de temps prend un import d'enregistrements basé sur la commande LECTURE ASCII :

```
   ` Méthode projet Import classique
$vhDocRef:=Ouvrir document("")
Si (OK=1)
   FERMER DOCUMENT($vhDocRef)
   FORMULAIRE ENTREE([Table1];"Import")
   $vhStartTime:=Heure courante
   LECTURE ASCII([Table1];Document)
   $vhEndTime:=Heure courante
   ALERTE("L'opération a duré "+Chaine(0+($vhEndTime-$vhStartTime))+" secondes.")
Fin de si
```

Avec l'import de données classique, 4D Client analyse le fichier ASCII puis, pour chaque enregistrement, crée un nouvel enregistrement, remplit les champs avec les valeurs importées et envoie l'enregistrement au poste serveur afin qu'il soit ajouté à la base. Par conséquent, de nombreuses requêtes circulent sur le réseau. Afin d'optimiser l'opération, vous pouvez utiliser des procédures stockées pour effectuer l'import localement sur le poste serveur. Le poste client charge le document dans un BLOB et déclenche une procédure stockée en passant le BLOB comme paramètre. La procédure stockée place le BLOB dans un document sur le disque du poste serveur, puis importe le document en local. L'import des données est ainsi effectué localement à une vitesse comparable à celle d'une version monoposte de 4D, car la plupart des requêtes transitant sur le réseau ont été éliminées. Voici la méthode projet CLIENT IMPORT. Lancée sur le poste client, elle déclenche l'exécution de la procédure stockée SERVER IMPORT, listée à la suite :

```
   ` Méthode projet CLIENT IMPORT
   ` CLIENT IMPORT ( Pointeur ; Alpha )
   ` CLIENT IMPORT ( -> [Table] ; Formulaire entrée )

C_POINTEUR($1)
C_ALPHA(31;$2)
C_HEURE($vhDocRef)
C_BLOB($vxData)
C_ENTIER LONG(spErrCode)
   ` Sélectionnez le document à importer
```

```
$vhDocRef:=Ouvrir document("")
Si (OK=1)
      ` Si un document était sélectionné, ne pas le garder ouvert
   FERMER DOCUMENT($vhDocRef)
   $vhStartTime:=Heure courante
      ` Essayons de le charger en mémoire
   DOCUMENT VERS BLOB(Document;$vxData)
   Si (OK=1)
         ` Si le document a pu être chargé dans le BLOB, démarrer la procédure
         ` stockée qui va importer les données sur le poste serveur
⇒     $spProcessID:=Executer sur serveur("SERVER IMPORT";32*1024;
                                    "Serveur Import Services";Table($1);$2;$vxData)
         ` Nous n'avons alors plus besoin du BLOB dans ce process
      EFFACER VARIABLE($vxData)
         ` Attendons l'achèvement de l'opération effectuée par la procédure stockée
      Repeter
         ENDORMIR PROCESS(Numero du process courant;300)
         LIRE VARIABLE PROCESS($spProcessID;spErrCode;spErrCode)
         Si (Indefinie(spErrCode))
               ` Note: si la procédure stockée n'a pas initialisé sa propre instance
               ` de la variable spErrCode, il se peut qu'une variable indéfinie soit
               ` retournée
            spErrCode:=1
         Fin de si
      Jusque (spErrCode<=0)
         ` Envoyons un accusé de réception à la procédure stockée
      spErrCode:=1
      ECRIRE VARIABLE PROCESS($spProcessID;spErrCode;spErrCode)
      $vhEndTime:=Heure courante
      ALERTE("L'opération a duré "+Chaine(0+($vhEndTime-$vhStartTime))+
                                                                  " secondes.")
   Sinon
      ALERTE("Il n'y a pas assez de mémoire pour charger le document.")
   Fin de si
Fin de si
```

Voici la méthode projet SERVER IMPORT, exécutée en tant que procédure stockée :

```
   ` Méthode projet SERVER IMPORT
   ` SERVER IMPORT ( Entier long ; Alpha ; BLOB )
   ` SERVER IMPORT ( Numéro de table ; Formulaire entrée ; Données importées )

C_ENTIER LONG($1)
C_ALPHA(31;$2)
C_BLOB($3)
C_ENTIER LONG(spErrCode)

   ` L'opération n'est pas encore terminée, affectons 1 à spErrCode
spErrCode:=1
$vpTable:=Table($1)
FORMULAIRE ENTREE($vpTable->;$2)
$vsDocName:="Fichier Import "+Chaine(1+Hasard)
SUPPRIMER DOCUMENT($vsDocName)
BLOB VERS DOCUMENT($vsDocName;$3)
LECTURE ASCII($vpTable->;$vsDocName)
SUPPRIMER DOCUMENT($vsDocName)
   ` L'opération est terminée, affectons 0 à spErrCode
spErrCode:=0
   ` Attendons que le poste client à l'origine de la requête ait reçu les résultats
Repeter
   ENDORMIR PROCESS(Numero du process courant;1)
Jusque (spErrCode>0)
```

Une fois que ces deux méthodes projet ont été implémentées dans votre base, vous pouvez effectuer un import basé sur une procédure stockée, en écrivant par exemple :

```
CLIENT IMPORT (->[Table1];"Import")
```

Si vous réalisez quelques tests comparatifs, vous pourrez constater qu'avec ce type de méthode, l'import des enregistrements est jusqu'à 60 fois plus rapide qu'un import "classique".

Référence

Nouveau process, Procédures stockées.

ENDORMIR PROCESS

Process

version 5

ENDORMIR PROCESS (process; durée)

Paramètre	Type		Description
process	Numérique	→	Numéro de process
durée	Numérique	→	Durée exprimée en ticks

Description

ENDORMIR PROCESS permet d'endormir un process pour un certain nombre de ticks (1 tick = 1/60ème de seconde). Pendant cette période, le process endormi n'utilise pas de temps machine. Il reste cependant toujours en mémoire.

Si le process est déjà endormi, cette commande l'endort à nouveau. Le paramètre durée n'est pas ajouté au temps restant mais le remplace. Vous pouvez passer zéro (0) dans durée si vous ne voulez plus endormir le process.

Si le process n'existe pas, la commande ne fait rien.

Attention : Utilisez ENDORMIR PROCESS uniquement avec les process que vous avez créés. ENDORMIR PROCESS n'affecte pas le process principal.

Astuce : Pour “endormir” le process principal (par exemple, pour afficher pendant un certain temps un message dans une fenêtre que vous ouvrez et fermez dans ce but), écrivez une petite sous-routine "d'attente" qui effectue une boucle pendant un certain temps à l'aide des fonctions Heure courante, Nombre de ticks ou Nombre de millisecondes).

Exemples

(1) Reportez-vous aux exemples de la section Verrouillage d'enregistrements.
(2) Reportez-vous à l'exemple de la fonction Chercher process.

Référence

CACHER PROCESS, SUSPENDRE PROCESS.

SUSPENDRE PROCESS

Process

version 5

SUSPENDRE PROCESS (process)

Paramètre	Type		Description
process	Numérique	→	Numéro de process

Description

SUSPENDRE PROCESS suspend l'exécution de process jusqu'à ce qu'il soit remis en action par la comande REACTIVER PROCESS. Pendant ce temps, process n'utilise pas de temps machine. Lorsqu'un process est suspendu, il existe toujours en mémoire.

Si process est déjà suspendu, SUSPENDRE PROCESS ne fait rien. Si le process est endormi à l'aide de ENDORMIR PROCESS, le process est suspendu. S'il reçoit l'ordre REACTIVER PROCESS, le process redevient actif immédiatement.

Lorsqu'un process est suspendu, les fenêtres qui lui appartiennent ne sont pas saisissables. Dans ce cas, si vous ne voulez pas dérouter l'utilisateur, il faut auparavant cacher le process. Si process n'existe pas, cette commande ne fait rien.

Attention : Utilisez SUSPENDRE PROCESS seulement avec les process que vous avez créés. SUSPENDRE PROCESS n'a aucun effet sur le process Utilisation/Menus créés.

Référence

CACHER PROCESS, ENDORMIR PROCESS, REACTIVER PROCESS.

REACTIVER PROCESS

version 5

REACTIVER PROCESS (process)

Paramètre	Type		Description
process	Numérique	→	Numéro de process

Description

REACTIVER PROCESS réactive un process suspendu ou endormi. Si process n'est pas endormi ou suspendu, REACTIVER PROCESS ne fait rien.

Si process a été suspendu, référez-vous aux commandes SUSPENDRE PROCESS ou ENDORMIR PROCESS. Si process n'existe pas, cette commande ne fait rien.

Référence

ENDORMIR PROCESS, SUSPENDRE PROCESS.

Process interrompu

Process

version 6.5

Process interrompu → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Vrai = le process est sur le point d'être interrompu, Faux = le process n'est pas sur le point d'être interrompu

Description

La commande Process interrompu retourne Vrai si le process dans lequel elle est appelée est sur le point d'être interrompu de manière inopinée — c'est-à-dire sans être parvenu au terme "normal" de son exécution. Cela peut se produire, par exemple, à la suite d'un appel à QUITTER 4D.

Exemple

Cette commande peut être utilisée dans le cadre d'un type particulier de programmation du serveur Web, en mode compilé uniquement : lorsque vous utilisez une méthode envoyant des pages Web à l'aide d'une boucle du type Tant que...Fin tant que, le mécanisme du serveur Web ne permet pas de stopper la boucle en cas de timeout (fin de période d'inactivité autorisée) avec un browser. Le process Web n'étant pas refermé, des ressources sont ainsi gaspillées.
La commande Process interrompu, placée dans le test initial de la boucle, retournera Vrai en cas de timeout. La boucle pourra donc être interrompue et le process tué.
Voici une méthode parfois utilisée pour envoyer des pages HTML. En mode compilé, cette boucle ne pourra pas être interrompue en cas de timeout :

```
Tant que (Vrai)
   ENVOYER FICHIER HTML (FichierHTML)
Fin tant que
```

La commande Process interrompu permet d'utiliser le même type de méthode, tout en préservant la possibilité de sortir de la boucle en cas de timeout :

```
⇒  Tant que (Non (Process interrompu))
      ENVOYER FICHIER HTML (FichierHTML)
   Fin tant que
```

Numero du process courant

Numero du process courant → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Numéro du process en cours d'exécution

Description

Numero du process courant retourne le numéro du process à partir duquel la fonction a été appelée.

Exemples

Référez-vous aux exemples de ENDORMIR PROCESS et INFORMATIONS PROCESS.

Référence

Chercher process, INFORMATIONS PROCESS, Statut du process.

Statut du process

Statut du process (process) → Numérique

Paramètre	Type		Description
process	Numérique	→	Numéro du process
Résultat	Numérique	←	Statut du process

Description

La commande Statut du process retourne le statut du process dont le numéro est passé dans process.

Le résultat de la fonction peut être l'une des valeurs des constantes prédéfinies suivantes :

Constante	Type	Valeur
Détruit	Entier long	-1
Endormi	Entier long	1
Inexistant	Entier long	-100
En exécution	Entier long	0
Dialogue caché	Entier long	6
Suspendu	Entier long	5
En attente entrée sortie	Entier long	3
En attente drapeau interne	Entier long	4
En attente événement	Entier long	2

Si le process n'existe pas (ce qui signifie le numéro que vous avez passé est hors de l'intervalle de 1 à Nombre de process), Statut du process retourne Inexistant (-100).

Exemple

L'exemple suivant retourne le nom et le numéro de référence du process de chaque process dans les tableaux asProcName et aiProcNum. La méthode teste si le process a été détruit. Dans ce cas, le nom et le numéro du process ne sont pas ajoutés dans le tableau :

```
$vlNbTasks:=Nombre de process
TABLEAU ALPHA(31;arProcName; $vlNbTasks)
TABLEAU ENTIER(aiProcNum; $vlNbTasks)
$vlActualCount:=0
Boucle ($vlProcess;1; $vlNbTasks)
⇒    Si (Statut du process($vlProcess)>=En exécution)
```

```
            $vlActualCount:=$vlActualCount+1
            INFORMATIONS PROCESS($vlProcess; asProcName{$vlActualCount};$vlState;
                                                $vlTime)aiProcNum{$vlActualCount}:=$vlProcess
        Fin de si
    Fin de boucle
        ` Eliminer les éléments superflus
    TABLEAU ALPHA(31;asProcName;$vlActualCount)
    TABLEAU ENTIER(aiProcNum;$vlActualCount)
```

Référence

INFORMATIONS PROCESS, Nombre de process.

INFORMATIONS PROCESS

Process

version 6.5 (Modifiée)

INFORMATIONS PROCESS (process; procNom; procStatut; procTemps{; procVisible{; uniqueID{; origine}}})

Paramètre	Type		Description
process	Numérique	→	Numéro du process
procNom	Alpha	←	Nom du process
procStatut	Numérique	←	Statut du process
procTemps	Numérique	←	Temps d'exécution cumulé du process en ticks
procVisible	Booléen	←	Visible (Vrai) ou Caché (Faux)
uniqueID	Entier	←	Numéro unique du process
origine	Entier long	←	Origine du process

Description

La commande INFORMATIONS PROCESS retourne les informations sur le process dont vous passez le numéro dans process.

Après l'appel :

• procNom retourne le nom du process. Quelques points sont à noter à propos du nom du process :

- Si le process a été démarré depuis la boîte de dialogue **Exécuter une méthode** en mode Utilisation (avec l'option **Nouveau process** sélectionnée), son nom est “P_” suivi d'un numéro.
- Si le process a été démarré à partir d'une commande de Menus créés dont la propriété **Démarrer un process** est sélectionnée, le nom du process est “M_” ou “ML_” suivi d'un numéro.
- Si un process a été stoppé (et son “espace” non encore réutilisé), son nom est encore retourné. Pour détecter si un process est stoppé, testez procStatut=-1 (voir ci-dessous).

• procStatut retourne le statut du process au moment de l'appel. Ce paramètre peut retourner l'une des valeurs fournies par les constantes prédéfinies suivantes (thème Statut du process) :

Constante	Type	Valeur
Détruit	Entier long	-1
Endormi	Entier long	1
Inexistant	Entier long	-100
En exécution	Entier long	0
Dialogue caché	Entier long	6
Suspendu	Entier long	5
En attente d'entrée sortie	Entier long	3
En attente d'un drapeau interne	Entier long	4
En attente d'événement	Entier long	2

• procTemps retourne le cumul du temps que le process a utilisé depuis qu'il a été démarré, en ticks (1/60e de seconde.)

• procVisible, s'il est spécifié, retourne VRAI si le process est visible, FAUX s'il est caché.

• uniqueID, s'il est spécifié, retourne le numéro unique du process. En effet, à compter de la version 6.5 de 4D, chaque process se voit attribuer un numéro de process ainsi qu'un numéro unique de process par session. Ce dernier permet de différencier strictement deux process ou sessions de process. Il correspond au nombre de process ayant été lancés au cours de la session de l'application 4e Dimension.

• origine, s'il est spécifié, retourne une valeur décrivant l'origine du process. Ce paramètre peut retourner l'une des valeurs fournies par les constantes prédéfinies suivantes (thème Type du process) :

Constante	Type	Valeur
Process web avec contexte	Entier long	-11
Autre process 4D	Entier long	-10
Tâche externe	Entier long	-9
Gestionnaire d'événement	Entier long	-8
Gestionnaire Apple Event	Entier long	-7
Gestionnaire du port série	Entier long	-6
Gestionnaire d'index	Entier long	-5
Gestionnaire du cache	Entier long	-4
Process web sans contexte	Entier long	-3
Process de structure	Entier long	-2
Process principal	Entier long	-1
Aucun	Entier long	0
Créé par programmation	Entier long	1
Créé par commande de menu	Entier long	2
Créé depuis le mode Utilisation	Entier long	3
Autre process utilisateur	Entier long	4

Note : Les process internes à 4D retournent une valeur négative et les process générés par l'utilisateur retournent une valeur positive.

Si le process n'existe pas, ce qui veut dire que vous n'avez pas passé un nombre inclus dans l'intervalle [1>Nombre de process], INFORMATIONS PROCESS laisse les valeurs des variables passées en paramètres inchangées.

Exemples

(1) L'exemple suivant retourne le nom, le statut, et le temps écoulé dans les variables vNom, vStatut, et vTempsPassé pour le process courant :

```
      C_ALPHA(80; vNom) ` Initialiser les variables
      C_ENTIER(vStatut)
      C_ENTIER(vTempsPassé)
⇒     INFORMATIONS PROCESS (Numero du process courant;vNom;vStatut;vTempsPassé)
```

(2) Voir l'exemple de la section Méthode base Sur fermeture.

Référence

Nombre de process, Statut du process.

Chercher process

Process

version 6.0

Chercher process (nom{; *}) → Numérique

Paramètre	Type		Description
nom	Alpha	→	Nom du process duquel récupérer le numéro
*	*	→	Retourner le numéro du process serveur
Résultat	Numérique	←	Numéro du process

Description

La commande Chercher process retourne le numéro du process dont vous passez le nom dans nom. Si aucun process n'est trouvé, Chercher process retourne 0.

Le paramètre optionnel * vous permet, à partir de 4D Client, de récupérer le numéro d'un process s'exécutant sur le serveur, c'est-à-dire une procédure stockée. Dans ce cas, la valeur retournée est négative. Cette option est particulièrement utile dans le cadre de l'utilisation des commandes LIRE VARIABLE PROCESS et ECRIRE VARIABLE PROCESS. Pour plus d'informations, reportez-vous à la description de ces commandes.
Si la commande est exécutée avec le paramètre * à partir d'un process tournant sur le poste serveur, la valeur retournée est positive.

Exemple

Vous créez une palette flottante, fonctionnant dans un process séparé, dans lequel vous implémentez vos propres outils pour interagir avec l'environnement Structure. Par exemple, quand vous sélectionnez un élément dans une liste hérarchique de mots-clés, vous voulez coller du texte dans la fenêtre de premier plan du mode Structure. Pour cela, vous pouvez utiliser le presse-papiers, mais l'événement de collage doit se passer dans le process Structure. La petite fonction qui suit retourne le numéro du process de Structure (s'il est actif) :

```
` Méthode projet Numéro process Structure
` Numéro process Structure  -> Entier long
` Numéro process Structure -> Numéro du process de Structure

⇒ $0:=Chercher process(Lire chaine dans liste(170;3))
  ` Le nom du process Structure est stocké dans la ressource 'STR#" ID=170, chaîne #3
  ` dans 4D
  ` Note: ceci peut ne pas fonctionner si la ressource est modifiée dans l'avenir
```

Avec cette fonction, la méthode projet listée ci-dessous colle le texte reçu en paramètre dans la fenêtre de premier plan du mode Structure (si c'est possible) :

```
` Méthode projet COLLER TEXTE EN STRUCTURE
` COLLER TEXTE EN STRUCTURE ( Texte)
` COLLER TEXTE EN STRUCTURE ( Texte à coller dans la fenêtre de Structure de
` premier plan )

C_TEXTE($1)
C_ENTIER LONG($vlStructurePID;$vlCompte)

$vlStructurePID:=Numero process Structure
Si ($vlStructurePID # 0)
      ` Mettre le texte dans le presse-papiers
   ECRIRE TEXTE DANS PRESSE PAPIERS($1)
      ` Générer un événement Ctrl-V / Command-V
   GENERER FRAPPE CLAVIER(Code ascii("v");Masque touche commande;
                                                        $vlStructurePID)
      ` Appeler répétitivement ENDORMIR PROCESS pour que le minuteur puisse passer
      ` l'événement au process Structure
   Boucle ($vlCompte;1;5)
      ENDORMIR PROCESS(Numero du process courant;1)
   Fin de boucle
Fin de si
```

Référence

INFORMATIONS PROCESS, Statut du process.

Nombre utilisateurs

Process

version 3

Nombre utilisateurs → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	Nombre d'utilisateurs connectés au serveur

Description

Nombre utilisateurs retourne le nombre d'utilisateurs connectés au poste serveur. Si le serveur exécute au moins une procédure stockée, Nombre utilisateurs retourne le nombre d'utilisateurs +1.

Dans le cas d'une version monoposte de 4e Dimension, Nombre utilisateurs retourne 1.

Référence

Nombre de process, Nombre de process utilisateurs.

Nombre de process

Process

version 5

Nombre de process → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	Nombre total de process ouverts (y compris les process du moteur de 4D)

Description

Nombre de process retourne le nombre de process ouverts sur un poste 4D Client ou dans une version monoposte de 4e Dimension.

Ce nombre inclut tous les process, qu'ils soient créés par vos soins ou par 4e Dimension. Les process créés automatiquement par 4D sont le Process principal, le Gestionnaire de cache, le process de Structure, le Process d'indexation et le Serveur Web.

La valeur retournée par Nombre de process comprend également les process qui ont été détruits.

Exemples

Référez-vous à l'exemple de Statut du process et Méthode base Sur fermeture.

Référence

INFORMATIONS PROCESS, Nombre de process utilisateurs, Nombre utilisateurs, Statut du process.

Nombre de process utilisateurs

version 5

Nombre de process utilisateurs → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	Nombre de process ouverts (à l'exception des process du moteur)

Description

Nombre de process utilisateurs retourne le nombre de process ouverts sur le serveur, à l'exception de ceux créés automatiquement par 4e Dimension.

Le "Process principal" (mode Utilisation/Menus créés) est considéré comme faisant partie des process utilisateurs. En outre, le process "Structure" et le "Process Serveur Web", pouvant être fermés par l'utilisateur, sont également considérés comme des process utilisateurs.
Ces trois process sont donc comptabilisés dans les process ouverts retournés par cette commande.
Par ailleurs, certains process sont comptabilisés comme process utilisateurs lorsqu'ils sont exécutés sur 4D Server en tant que procédure stockée.

Le tableau suivant résume ce fonctionnement :

Process	4D	4D Client	4D Server (Proc. stockée)
Process principal	X	X	X
Structure	X	X	X
Serveur Web	X	X	X
Gestionnaire client	-	0	X
Interface utilisateur	-	0	X
Procédure stockée	-	-	X

X = process inclus dans les process utilisateurs
0 = process non inclus dans les process utilisateurs

Référence

Nombre de process, Nombre utilisateurs.

EXECUTER SUR CLIENT

Process

version 6.5

EXECUTER SUR CLIENT (nomClient; nomMéthode{; param}{; param2; ...; paramN})

Paramètre	Type		Description
nomClient	Alpha	→	Nom d'inscription du 4D Client
nomMéthode	Alpha	→	Nom de la méthode à exécuter
param		→	Paramètre(s) de la méthode

Description

La commande EXECUTER SUR CLIENT provoque l'exécution de la méthode nomMéthode, avec, éventuellement, le(s) paramètre(s) param1... paramN, sur le ou les 4D Client inscrit(s) sous le nom nomClient. Le nom d'inscription du ou des 4D Client est défini par la commande INSCRIRE CLIENT.

Cette commande peut être appelée depuis un 4D Client ou une procédure stockée sur 4D Server.
Si la méthode admet des paramètres, passez-les après le nom de la méthode.

L'exécution de la méthode sur le 4D Client s'effectue dans un process global créé automatiquement sur le poste client, et portant le nom d'inscription du 4D Client.

Si cette commande est appelée plusieurs fois de suite pour un même 4D Client, les ordres d'exécution seront empilés. Par conséquent, les méthodes seront traitées les unes à la suite des autres : les exécutions sont asynchrones. Plus l'empilement est grand, plus la "charge de travail" est grande pour le 4D Client. Vous pouvez connaître l'état de la charge de travail de chaque client à l'aide de la commande LIRE CLIENTS INSCRITS.

Note : L'empilement des ordres d'exécutions ne peut être modifié ou stoppé, sauf si le 4D Client est désinscrit à l'aide de la commande DESINSCRIRE CLIENT.

Il est possible d'exécuter simultanément la même méthode sur plusieurs ou sur la totalité des 4D Clients inscrits : pour cela, passez le caractère joker (@) dans le paramètre nomClient.

Exemples

(1) Vous souhaitez exécuter sur le poste client "Client1" la méthode "GénéreNums", comportant trois paramètres :

```
⇒ EXECUTER SUR CLIENT("Client1";"GénéreNums";12;$a;"Text")
```

(2) Vous souhaitez que tous les clients inscrits exécutent la méthode "VideTemp" :

```
⇒ EXECUTER SUR CLIENT("@";"VideTemp")
```

(3) Reportez-vous à l'exemple de la commande INSCRIRE CLIENT.

Référence

DESINSCRIRE CLIENT, INSCRIRE CLIENT, LIRE CLIENTS INSCRITS.

Variables et ensembles système

La variable système *OK* prend la valeur 1 si 4D Server a correctement reçu la requête d'exécution d'une méthode — cela ne garantit pas toutefois la bonne exécution de la méthode sur le 4D Client.

INSCRIRE CLIENT

Process

version 6.5

INSCRIRE CLIENT (nomClient{; période{; *}})

Paramètre	Type		Description
nomClient	Alpha	→	Nom de la session 4D Client
période	Entier long	→	Délai d'interrogation au serveur (en secondes)
*	*	→	Process local

Description

La commande INSCRIRE CLIENT "inscrit" un poste 4D Client sous le nom nomClient auprès de 4D Server, afin de permettre que d'autres clients ou éventuellement 4D Server (par l'intermédiaire de procédures stockées) puissent y exécuter des méthodes à l'aide de la commande EXECUTER SUR CLIENT. Une fois inscrit, un 4D Client peut donc exécuter une ou plusieurs méthodes pour le compte d'autres clients.

Note : Vous pouvez également inscrire automatiquement chaque poste client qui se connecte à 4D Server, dans la boîte de dialogue des Propriétés de la base (cf. manuel *Mode Structure*).

A l'issue de l'exécution de la commande, un process, nommé nomClient, est créé sur le poste client. Ce process ne peut être détruit que par la commande DESINSCRIRE CLIENT. Si le paramètre optionnel * est passé, le process créé est local (4D ajoute automatiquement le signe $ au nom du process). Sinon, il est global.

Note : Plusieurs 4D Client peuvent avoir le même nom d'inscription.

Après l'exécution de la commande, le poste client interroge périodiquement 4D Server afin de savoir si un autre 4D Client ou le serveur lui-même font appel à lui. Par défaut, cette interrogation est effectuée toutes les 2 secondes. Vous pouvez modifier ce délai à l'aide du paramètre période. La valeur minimale est de 1 seconde.

Note : Lorsqu'elle est utilisée avec 4e Dimension monoposte, cette commande ne fait rien.

Une fois la commande exécutée, il n'est pas possible de modifier "à la volée" le nom du 4D Client ni le délai d'interrogation du serveur. Pour cela, il est nécessaire d'appeler la commande DESINSCRIRE CLIENT puis d'exécuter à nouveau INSCRIRE CLIENT.

Si le 4D Client est correctement inscrit, la variable système *OK* prend la valeur 1. Si le 4D Client était déjà inscrit, la commande ne fait rien et *OK* prend la valeur 0.

Exemple

Les méthodes suivantes permettent de réaliser une petite messagerie entre les postes clients inscrits.

1. La méthode *INSCRIPTION* permet d'inscrire un 4D Client et de le tenir prêt à recevoir un message de la part d'un autre 4D Client :

```
      `Méthode INSCRIPTION
      `Il faut se désinscrire avant de s'inscrire sous un autre nom
⇒     DESINSCRIRE CLIENT
      Repeter
         vNomPseudo:=Demander("Entrez votre nom :";"Utilisateur";"OK";"Annuler")
      Jusque ((OK=0) | (vNomPseudo # ""))
      Si (OK=0)
         ...` Ne rien faire
      Sinon
⇒        INSCRIRE CLIENT(vNomPseudo;*)
      Fin de si
```

2. L'instruction suivante permet de connaître les 4D Clients inscrits. Elle peut être placée dans la Méthode base Sur ouverture :

```
      ` Méthode base Sur ouverture
   PrListeClient:=Nouveau process("Liste_4DClients";32000;"Liste d'inscrits")
```

3. La méthode *Liste_4DClients* permet de récupérer tous les 4D Client inscrits et les personnes acceptant de recevoir des messages :

```
      ` Méthode Liste_4DClients
   Si (Type application=4D Client)
         ` Le code ci-dessous n'est valable qu'en mode client-serveur
      $Ref:=Creer fenetre(100;100;300;400;-(Fenêtre palette+Avec titre de fenêtre);
                                                                  "Liste d'inscrits")
      Repeter
⇒        LIRE CLIENTS INSCRITS($ListeClient;$ListeCharge)
            `Récupération des clients inscrits dans $ListeClient
         EFFACER FENETRE($Ref)
         POSITION MESSAGE(0;0)
         Boucle ($p;1;Taille tableau($ListeClient))
            MESSAGE($ListeClient{$p}+Caractere(Retour chariot))
         Fin de boucle
            `Afficher chaque seconde
         ENDORMIR PROCESS(Numero du process courant;60)
      Jusque (Faux) ` Boucle infinie
   Fin de si
```

4. La méthode *Envoyer_Message* permet d'envoyer un message à un autre 4D Client inscrit.

```
    ` Méthode Envoyer_Message
$Destinataire:=Demander("Destinataire du message :";"")
    ` Saisir le nom d'une des personnes visibles dans la fenêtre générée par la
    ` méthode base Sur ouverture
Si (OK # 0)
    $Message:=Demander("Message :")   ` Contenu du message
    Si (OK # 0)
⇒       EXECUTER SUR CLIENT($Destinataire;"Afficher_Message";$Message)
            ` Envoi du message
    Fin de si
Fin de si
```

5. La méthode *Afficher_Message* affiche le message sur le poste client :

```
    ` Méthode Afficher_Message
C_TEXTE($1)
ALERTE($1)
```

6. Enfin, cette méthode permet à un poste client de n'être plus visible par les autres 4D Clients et ne plus recevoir de message :

```
    ` Méthode DÉSINSCRIPTION :
⇒   DESINSCRIRE CLIENT
```

Référence

DESINSCRIRE CLIENT, EXECUTER SUR CLIENT, LIRE CLIENTS INSCRITS.

Variables et ensembles système

Si le poste client est correctement inscrit, la variable système *OK* prend la valeur 1. Si le poste était déjà inscrit, la commande ne fait rien et *OK* prend la valeur 0.

DESINSCRIRE CLIENT

Process

version 6.5

DESINSCRIRE CLIENT

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande DESINSCRIRE CLIENT “désinscrit” le poste 4D Client de 4D Server. Il doit avoir été préalablement inscrit à l’aide de la commande INSCRIRE CLIENT.

Si le poste client n’était pas inscrit ou si la commande est exécutée sur 4e Dimension monoposte, la commande ne fait rien.

Note : Un 4D Client est automatiquement désinscrit lorsque l’application quitte.

Exemple

Reportez-vous à l’exemple de la commande INSCRIRE CLIENT.

Référence

EXECUTER SUR CLIENT, INSCRIRE CLIENT, LIRE CLIENTS INSCRITS.

Variables et ensembles système

Si le client est correctement désinscrit, la variable système *OK* prend la valeur 1. Si le 4D Client n’était pas inscrit, *OK* prend la valeur 0.

LIRE CLIENTS INSCRITS

Process

version 6.5

LIRE CLIENTS INSCRITS (listeClients; nbMéthodes)

Paramètre	Type		Description
listeClients	Tab Texte	←	Liste des 4D Client enregistrés
nbMéthodes	Tab Entier long	←	Liste des méthodes restant à exécuter

Description

La commande LIRE CLIENTS INSCRITS remplit deux tableaux :

• listeClients, qui contient la liste des clients “inscrits” à l’aide de la commande INSCRIRE CLIENT.

• nbMéthodes, qui fournit liste des “charges de travail” de chaque client. La charge de travail est le nombre de méthodes qu’un 4D Client doit encore exécuter, à la demande de la commande EXECUTER SUR CLIENT.

Exemples

(1) Vous souhaitez obtenir la liste des clients inscrits et des méthodes restant à exécuter :

```
     TABLEAU TEXTE($clients;0)
     TABLEAU ENTIER LONG($nprocs;0)
⇒    LIRE CLIENTS INSCRITS($clients;$nprocs)
```

2) Reportez-vous à l’exemple de la commande INSCRIRE CLIENT.

Référence

DESINSCRIRE CLIENT, EXECUTER SUR CLIENT, INSCRIRE CLIENT.

Variables et ensembles système

Si l’opération se déroule correctement, la variable système *OK* prend la valeur 1.

40

Process (Communications)

Semaphore (sémaphore{; nbTicks}) → Booléen

Paramètre	Type		Description
sémaphore	Alpha	→	Sémaphore à tester et à positionner
nbTicks	Entier	→	Temps d'attente maximum
Résultat	Booléen	←	sémaphore a été correctement créé (Faux) ou sémaphore était déjà créé (Vrai)

Description

Un sémaphore est un drapeau visible par chaque poste client (l'ordinateur de chaque utilisateur) ou chaque process sur un même poste. Un sémaphore a simplement pour rôle d'exister ou de ne pas exister. Chaque méthode exécutée par un utilisateur peut tester la présence d'un sémaphore. En créant et en testant des sémaphores, vous permettez aux méthodes de communiquer entre les postes clients et les process.

La fonction Semaphore retourne Vrai si sémaphore existe. Si sémaphore n'existe pas, Semaphore le crée et retourne Faux. Un seul utilisateur à la fois peut créer un sémaphore. Si Semaphore retourne Faux, cela indique que sémaphore n'existait pas, mais cela signifie également que sémaphore a été créé et positionné dans le process d'où l'appel a été effectué.

Semaphore retourne Faux si le sémaphore n'existait pas. La fonction retourne également Faux si le sémaphore avait été déjà positionné par le process d'où l'appel a été effectué. Un sémaphore est limité à 30 caractères, métacaractères ($, <>) inclus. Si vous passez une chaîne plus longue, elle est tronquée.

Le paramètre optionnel nbTicks vous permet de spécifier un délai d'attente en ticks (1 tick = 1/60ème de seconde) si sémaphore est déjà positionné. Dans ce cas, avant de retourner Vrai, la fonction attend, dans la limite du temps fixé, que sémaphore se libère (auquel cas elle retourne Faux). Si le délai expire sans que sémaphore ait été libéré, Semaphore retourne Vrai.

Il y a deux types de sémaphores dans 4e Dimension : les sémaphores **locaux** et les sémaphores **globaux**.

Un sémaphore local est visible par tous les process d'un même poste et seulement sur ce poste. Vous déclarez un sémaphore local en préfixant son nom avec le signe dollar ($). Les sémaphores locaux permettent de contrôler des opérations entre les différents process exécutés sur le même poste. Par exemple, un sémaphore local peut être utilisé pour gérer les accès à un tableau interprocess appelé par tous les process d'une base de données mono-utilisateur ou d'un poste client.
Un sémaphore global est visible par tous les utilisateurs et tous les process. Les sémaphores globaux permettent de contrôler des opérations entre les postes clients d'une base multi-utilisateurs.

Le principe de fonctionnement des sémaphores globaux et locaux est identique. Leur différence réside uniquement dans leur portée, c'est-à-dire leur visibilité. Dans 4D Server, les sémaphores globaux sont visibles pour tous les process de tous les postes clients. Un sémaphore local n'est visible que pour les process du poste client sur lequel il a été créé.

Avec 4e Dimension (monoposte), les sémaphores globaux et locaux ont la même portée car il n'y a qu'un seul utilisateur. Cependant, si votre base est utilisée dans les deux environnements, n'hésitez pas à employer des sémaphores globaux et locaux, en fonction de vos besoins.

Les sémaphores ne servent pas à protéger l'accès aux enregistrements — cette gestion est effectuée automatiquement par 4e Dimension et 4D Server. Les sémaphores ont pour but d'éviter que plusieurs utilisateurs effectuent la même opération en même temps.

Exemples

(1) Dans l'exemple suivant, vous souhaitez empêcher que deux utilisateurs effectuent simultanément une mise à jour globale des prix dans une table [Produits]. Pour cela, des sémaphores sont utilisés :

```
⇒   Si (Semaphore("MAJPrix"))  ` Essai de création du sémaphore
       ALERTE("Un autre utilisateur est déjà en train de mettre à jour les prix. Essayez plus
                                                                                    tard.")
    Sinon
       MAJdesPrix  ` Méthode de mise à jour des prix
       EFFACER SEMAPHORE("MAJPrix"))  ` Effacer le sémaphore
    Fin de si
```

(2) L'exemple suivant illustre l'utilisation d'un sémaphore local. Dans une base comportant plusieurs process, vous souhaitez maintenir une liste de "Choses à faire". Vous envisagez de la maintenir à jour dans un tableau interprocess et non dans une table. Vous devez empêcher les accès simultanés à l'aide d'un sémaphore. Dans ce cas, il vous suffit d'utiliser un sémaphore local car la liste "Choses à faire" est pour votre utilisation personnelle.

Le tableau interprocess est initialisé dans la méthode base Sur ouverture :

```
TABLEAU TEXTE (<>ListeAFaire;0) ` La liste de choses à faire est vide
```

Voici la méthode utilisée pour ajouter des éléments à la "liste des choses à faire" :

```
` Méthode projet AJOUTER LISTE A FAIRE
` AJOUTER LISTE A FAIRE (Texte)
` AJOUTER LISTE A FAIRE (Elément la liste à faire)

C_TEXTE($1)  ` Paramètre passé à la commande
⇒ Si (Non (Semaphore("$AccèsListe";300)))  ` Attendre 5 secondes maximum
   $vlElem:=Taille tableau(<>ListeAFaire)+1
   INSERER LIGNES(<>ListeAFaire;$vlElem)
   <>ListeAFaire{$vlElem}:=1
   EFFACER SEMAPHORE("$AccèsListe")  ` Effacer le sémaphore
Fin de si
```

Vous pouvez appeler cette méthode depuis n'importe quel process.

Référence

EFFACER SEMAPHORE, Tester semaphore.

EFFACER SEMAPHORE (sémaphore)

Paramètre	Type		Description
sémaphore	Alpha	→	Sémaphore à effacer

Description

EFFACER SEMAPHORE permet d'effacer le sémaphore précédemment créé par la fonction Semaphore.

La règle d'utilisation est que tous les sémaphores doivent être effacés lorsqu'ils ne sont plus nécessaires. Si les sémaphores ne sont pas effacés, ils restent en mémoire jusqu'à la fermeture du process dans lequel ils ont été créés.
Un process ne peut effacer que les sémaphores qu'il a créés. Si vous tentez d'effacer un sémaphore depuis un autre process que celui qui l'a créé, EFFACER SEMAPHORE ne fait rien.

Exemple

Reportez-vous à l'exemple de la fonction Semaphore.

Référence

Semaphore.

Tester semaphore

Process (Communications)

version 6.5

Tester semaphore (sémaphore) → Booléen

Paramètre	Type		Description
sémaphore	Alpha	→	Nom du sémaphore à tester
Résultat	Booléen	←	Vrai = le sémaphore existe, Faux = le sémaphore n'existe pas

Description

La commande Tester semaphore permet de tester l'existence d'un sémaphore.

A la différence de la fonction Semaphore, Tester semaphore ne crée pas le sémaphore s'il n'existe pas.
Si le sémaphore existe, la fonction retourne Vrai, s'il n'existe pas elle retourne Faux.

Exemple

Cet exemple permet de connaître l'état d'un traitement (en l'occurrence, la modification d'un code) sans modifier le sémaphore :

```
      Creer fenetre (x1;x2;y1;y2;-Fenêtre palette)
      Repeter
⇒        Si (Tester semaphore("Code d'encryptage"))
            POSITION MESSAGE ($x3;$y3)
            MESSAGE("Code d'encryptage en cours de modification.")
         Sinon
            POSITION MESSAGE($x3;$y3)
            MESSAGE("Modification du code d'encryptage autorisée.")
         Fin de si
      Jusque (StopInfo)
      FERMER FENETRE
```

Référence

Semaphore.

APPELER PROCESS (process)

Paramètre	Type		Description
process	Numérique	→	Numéro du process

Description

APPELER PROCESS appelle le formulaire affiché dans la fenêtre au premier plan de process.

Important : APPELER PROCESS ne fonctionne qu'avec des process tournant sur la même machine.

Si vous appelez un process qui n'existe pas, la commande ne fait rien.

Si process (le process appelé) n'a aucune fenêtre ou si aucun formulaire n'est affiché, rien ne se passe. Le formulaire affiché dans le process appelé reçoit un événement Sur appel extérieur. Cet événement doit avoir été sélectionné pour le formulaire dans la fenêtre des **propriétés de formulaire** en mode Structure, et vous devez le traiter dans la méthode formulaire. Si l'événement n'est pas sélectionné ou géré dans la méthode formulaire, la commande ne fait rien.

Le process appelant (dans lequel la commande APPELER PROCESS est exécutée) n'attend pas : APPELER PROCESS a un effet immédiat. Il est de votre ressort d'écrire, si nécessaire, une boucle d'attente pour traiter une éventuelle réponse du process appelé à l'aide des variables interprocess ou des variables process (réservées à cette utilisation) pouvant être lues et écrites entre les deux process avec les commandes LIRE VARIABLE PROCESS et ECRIRE VARIABLE PROCESS.

Si vous voulez établir une communication entre des process qui n'affichent pas de formulaires, utilisez les commandes LIRE VARIABLE PROCESS et ECRIRE VARIABLE PROCESS.

APPELER PROCESS accepte la syntaxe alternative APPELER PROCESS (-1).
Pour ne pas ralentir l'exécution d'une méthode, 4e Dimension ne redessine pas les variables interprocess à chaque fois qu'elles sont modifiées. Si vous passez -1 au lieu du numéro du process dans le paramètre process de la commande APPELER PROCESS, toutes les variables interprocess affichées dans toutes les fenêtres de tous les process seront mises à jour et redessinées.

Exemple

Reportez-vous à l'exemple de la section Méthode base Sur fermeture.

Référence

ECRIRE VARIABLE PROCESS, Evenement formulaire, LIRE VARIABLE PROCESS.

LIRE VARIABLE PROCESS

Process (Communications)

version 6.0

LIRE VARIABLE PROCESS (process; varSource; varDestination{; varSource2; varDestination2; ...; varSourceN; varDestinationN})

Paramètre	Type		Description
process	Numérique	→	Numéro de process source
varSource	Variable	→	Variable source
varDestination	Variable	←	Variable de destination

Description

La commande LIRE VARIABLE PROCESS lit la valeur de la ou des variable(s) process varSource (varSource2, etc.) depuis le process source dont le numéro est passé dans process et la retourne dans la ou les variables(s) varDestination (varDestination2, etc.) du process courant.

Chaque variable source peut être une variable, un tableau ou un élément de tableau. Tenez cependant compte des restrictions évoquées plus bas.

Pour chaque association varSource;varDestination les types des deux variables doivent être compatibles, sinon vous pourrez obtenir des valeurs non significatives.

Le process courant "pille" les variables du process de destination : ce dernier n'est averti en aucune manière de la lecture de l'instance de ses variables par un autre process.

4D Server : A partir d'un 4D Client, vous pouvez lire des variables dans un process de destination exécuté sur le poste serveur (procédure stockée). Pour cela, passez dans process le numéro du process serveur en négatif, c'est-à-dire précédé du signe - (moins). Attention, la communication process “intermachine” permise par les commandes LIRE VARIABLES, ECRIRE VARIABLE PROCESS et VARIABLE VERS VARIABLE n’est possible que du client vers le serveur. C’est toujours un process client qui lit ou écrit les variables d’une procédure stockée.

Astuce : Si vous ne connaissez pas le numéro du process serveur source, vous pouvez tout de même lire les variables interprocess du serveur. Pour cela, il vous suffit de passer toute valeur négative dans process. En d'autres termes, il n'est pas nécessaire de connaître précisément le numéro d'un process exécuté sur le serveur pour utiliser LIRE VARIABLE PROCESS avec les variables interprocess du serveur.
Cette possibilité s'avère particulièrement utile dans le cas d'une procédure stockée lancée sur le serveur par l'intermédiaire de la Méthode base Sur démarrage serveur. Comme, par défaut, les postes clients ne connaissent pas le numéro de ce process serveur, il vous suffit de passer une valeur négative (n'importe laquelle) dans le paramètre process.

Restrictions
LIRE VARIABLE PROCESS n'accepte pas de variables locales comme variables sources. En revanche, les variables de destination peuvent être interprocess, process ou locales. Vous pouvez "recevoir" les valeurs uniquement dans des variables, pas dans des champs.

LIRE VARIABLE PROCESS accepte tout type de variable source, process ou interprocess, à l'exception des variables de type :
• Pointeur
• Tableau de pointeurs
• Tableau à deux dimensions

Le process source doit être un process utilisateur, ce ne peut être un des process du moteur de 4D. Si le process source n'existe pas, la commande ne fait rien.

Note : En mode interprété, si une variable source n'existe pas, la valeur indéfinie est retournée. Vous pouvez le détecter en testant la variable de destination correspondante à l'aide de la fonction Type. En mode compilé, si aucune variable n'est associée au process source, une erreur est retournée. Vous pouvez intercepter cette erreur à l'aide d'une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Exemples
(1) La ligne de code suivante lit la valeur de la variable Texte vtCurStatus dans le process dont le numéro est $vlProcess et retourne le résultat dans la variable process vtInfo du process courant :

⇒ **LIRE VARIABLE PROCESS**($vlProcess;vtCurStatus;vtInfo)

(2) La ligne de code suivante fait la même chose mais retourne la valeur dans la variable locale $vtInfo de la méthode s'exécutant dans le process courant :

⇒ **LIRE VARIABLE PROCESS**($vlProcess;vtCurStatus;$vtInfo)

(3) La ligne de code suivante fait la même chose mais retourne la valeur dans la même variable vtCurStatus du process courant :

⇒ **LIRE VARIABLE PROCESS**($vlProcess;vtCurStatus;vtCurStatus)

Note : La première vtCurStatus désigne l'instance de la variable dans le process source, la seconde vtCurStatus désigne l'instance de la variable dans le process courant.

(4) L'exemple suivant lit séquentiellement les éléments d'un tableau process depuis le process indiqué par $vlProcess :

```
LIRE VARIABLE PROCESS($vlProcess;vl_IPCom_Array;$vlSize)
Boucle($vlElem;1;$vlSize)
⇒    LIRE VARIABLE PROCESS($vlProcess;at_IPCom_Array{$vlElem};$vtElem)
        ` Faire quelque chose avec $vtElem
Fin de boucle
```

Note : Dans cet exemple, la variable process vl_IPCom_Array doit être gérée par le process source et contient la taille du tableau at_IPCom_Array.

(5) L'exemple suivant fait la même chose que le précédent mais lit le tableau dans son intégralité au lieu de le faire élément par élément :

```
⇒    LIRE VARIABLE PROCESS($vlProcess;at_IPCom_Array;$anArray)
     Boucle($vlElem;1;Taille tableau($anArray))
        ` Faire quelque chose avec $anArray{$vlElem}
     Fin de boucle
```

(6) L'exemple suivant lit l'instance des variables v1,v2,v3 dans le process source et retourne leurs valeurs dans l'instance des mêmes variables du process courant :

```
⇒    LIRE VARIABLE PROCESS($vlProcess;v1;v1;v2;v2;v3;v3)
```

(7) Reportez-vous à l'exemple de la commande PROPRIETES GLISSER DEPOSER.

Référence

APPELER PROCESS, ECRIRE VARIABLE PROCESS, Introduction aux process, Présentation du Glisser-Déposer, PROPRIETES GLISSER DEPOSER, VARIABLE VERS VARIABLE.

ECRIRE VARIABLE PROCESS

Process (Communications)

version 6.0

ECRIRE VARIABLE PROCESS (process; varDestination; exprSource{; varDestination2; exprSource2; ...; varDestinationN; exprSourceN})

Paramètre	Type		Description
process	Numérique	→	Numéro de process de destination
varDestination	Variable	→	Variable de destination
exprSource	Variable	→	Expression source (ou variable source)

Description

La commande ECRIRE VARIABLE PROCESS écrit la ou les valeur(s) de exprSource (exprSource2, etc.) dans la ou les variable(s) process varDestination (varDestination2, etc.) du process de destination dont le numéro est passé dans process.
Chaque variable de destination peut être une variable ou un élément de tableau. Tenez cependant compte des restrictions évoquées ci-dessous.

Pour chaque association varDestination;exprSource, le type de l'expression doit être compatible avec la variable de destination, sinon vous pourrez obtenir des variables avec des valeurs incorrectes. En mode interprété, si la variable de destination n'existe pas, elle est créée et reçoit l'expression. En mode compilé, si aucune variable n'est associée au process de destination, une erreur est retournée. Vous pouvez intercepter cette erreur à l'aide d'une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Lorsque le process courant écrit les variables du process de destination, ce dernier n'est averti en aucune manière de l'écriture de l'instance de ses variables par un autre process.

4D Server : A partir d'un 4D Client, vous pouvez écrire des variables dans un process de destination exécuté sur le poste serveur (procédure stockée). Pour cela, passez dans process le numéro du process serveur en négatif, c'est-à-dire précédé du signe - (moins).
Attention, la communication process “intermachine” permise par les commandes LIRE VARIABLES, LIRE VARIABLE PROCESS et VARIABLE VERS VARIABLE n’est possible que du client vers le serveur. C’est toujours un process client qui lit ou écrit les variables d’une procédure stockée.

Astuce : Si vous ne connaissez pas le numéro du process serveur de destination, vous pouvez tout de même écrire dans les variables interprocess du serveur. Pour cela, il vous suffit de passer toute valeur négative dans process. En d'autres termes, il n'est pas nécessaire de connaître précisément le numéro d'un process exécuté sur le serveur pour utiliser ECRIRE VARIABLE PROCESS avec des variables interprocess du serveur.
Cette possibilité s'avère particulièrement utile dans le cas d'une procédure stockée lancée sur le serveur par l'intermédiaire de la Méthode base Sur démarrage serveur. Comme les postes clients ne connaissent pas forcément le numéro de ce process serveur, il vous suffit de passer une valeur négative (n'importe laquelle) dans le paramètre process.

Restrictions

ECRIRE VARIABLE PROCESS n'accepte pas de variables locales comme variables de destination.

ECRIRE VARIABLE PROCESS accepte tout type de variable process ou interprocess de destination, à l'exception :
• des variables de type Pointeur.
• des tableaux de tous types. Pour écrire un tableau entier d'un process vers un autre, utilisez la commande VARIABLE VERS VARIABLE. Notez cependant que ECRIRE VARIABLE PROCESS vous permet d'écrire des éléments de tableaux.
• des éléments de tableaux de pointeurs et des éléments de tableaux à deux dimensions.

Le process de destination doit être un process utilisateur, ce ne peut être un des process du moteur de 4D. Si le process de destination n'existe pas, la commande ne fait rien.

Exemples

(1) La ligne de code suivante affecte une chaîne vide à la variable Texte vtCurStatus du process dont le numéro est $vlProcess :

⇒ **ECRIRE VARIABLE PROCESS**($vlProcess;vtCurStatus;"")

(2) La ligne de code suivante affecte la variable Texte vtCurStatus du process dont le numéro est $vlProcess à la valeur de la variable $vtInfo depuis la méthode en cours d'exécution du process courant :

⇒ **ECRIRE VARIABLE PROCESS**($vlProcess;vtCurStatus;$vtInfo)

(3) La ligne de code suivante affecte la variable Texte vtCurStatus du process dont le numéro est $vlProcess à la valeur de la même variable dans le process courant :

⇒ **ECRIRE VARIABLE PROCESS**($vlProcess;vtCurStatus;vtCurStatus)

Note : Le premier vtCurStatus désigne l'instance de la variable dans le process de destination, le second vtCurStatus désigne l'instance de la variable dans le process courant.

(4) L'exemple suivant place séquentiellement en majuscules les éléments d'un tableau process depuis le process désigné par $vlProcess:

```
LIRE VARIABLE PROCESS($vlProcess;vl_IPCom_Array;$vlSize)
Boucle($vlElem;1;$vlSize)
   LIRE VARIABLE PROCESS($vlProcess;at_IPCom_Array{$vlElem};$vtElem)
⇒  ECRIRE VARIABLE PROCESS($vlProcess;at_IPCom_Array{$vlElem};Majusc($vtElem))
Fin de boucle
```

Note : Dans cet exemple, la variable process vl_IPCom_Array doit être gérée par les process source/destination et contient la taille du tableau at_IPCom_Array.

(5) L'exemple suivant écrit l'instance des variables v1, v2, v3 dans le process de destination à partir de l'instance de ces mêmes variables dans le process courant :

```
⇒  ECRIRE VARIABLE PROCESS($vlProcess;v1;v1;v2;v2;v3;v3)
```

Référence

APPELER PROCESS, Introduction aux process, LIRE VARIABLE PROCESS, VARIABLE VERS VARIABLE.

VARIABLE VERS VARIABLE (process; varDestination; varSource{; varDestination2; varSource2; ...; varDestinationN; varSourceN})

Paramètre	Type		Description
process	Numérique	→	Numéro du process de destination
varDestination	Variable	→	Variable de destination
varSource	Variable	→	Variable source

Description

La commande VARIABLE VERS VARIABLE écrit la valeur de la ou des variable(s) varSource1 (varSource2, etc.), dans la ou les variable(s) process varDestination (varDestination2, etc.) du process de destination dont vous avez passé le numéro dans process.

VARIABLE VERS VARIABLE a un fonctionnement semblable à celui de la commande ECRIRE VARIABLE PROCESS, avec cependant les différences suivantes :
• Alors que vous passez comme source à ECRIRE VARIABLE PROCESS des expressions (et donc vous ne pouvez pas passer un tableau en totalité), vous devez passer comme source à VARIABLE VERS VARIABLE uniquement des variables (et donc vous pouvez passer un tableau en totalité).
• Avec ECRIRE VARIABLE PROCESS, chaque variable de destination peut être une variable ou un élément de tableau, mais ne peut pas être un tableau. Avec VARIABLE VERS VARIABLE, chaque variable de destination peut être une variable, un tableau ou un élément de tableau.

4D Server : La communication process “intermachine” permise par les commandes VARIABLE VERS VARIABLE, ECRIRE VARIABLE PROCESS et LIRE VARIABLE PROCESS n’est possible que du client vers le serveur. C’est toujours un process client qui lit ou écrit les variables d’une procédure stockée.

Pour chaque association varDestination;varSource, le type de la variable source doit être compatible avec la variable de destination, sinon vous pourrez obtenir des variables avec des valeurs non significatives. En mode interprété, si la variable de destination n'existe pas, elle est créée puis le type et la valeur de la variable source lui sont affectés.

Lorsque le process courant écrit les variables du process de destination, ce dernier n'est averti en aucune manière de l'écriture de l'instance de ses variables par un autre process.

Restrictions

VARIABLE VERS VARIABLE n'accepte pas de variables locales comme variables de destination.

VARIABLE VERS VARIABLE accepte tout type de variable process ou interprocess de destination, à l'exception de variables de type :
• Pointeur
• Tableau de pointeurs
• Tableau à deux dimensions

Le process de destination doit être un process utilisateur, ce ne peut être un des process du moteur de 4D. Si le process de destination n'existe pas, une erreur est retournée. Vous pouvez intercepter cette erreur à l'aide d'une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Exemple

L'exemple suivant récupère un tableau process depuis le process désigné par $vlProcess, passe séquentiellement tous ses éléments en caractères majuscules puis réécrit entièrement le tableau :

```
   LIRE VARIABLE PROCESS($vlProcess;at_IPCom_Tab;$anTab)
   Boucle($vlElem;1;Taille tableau($anTab))
      $anTab{$vlElem}:=Majusc($anTab{$vlElem})
   Fin de boucle
⇒  VARIABLE VERS VARIABLE($vlProcess;at_IPCom_Tab;$anTab)
```

Référence

ECRIRE VARIABLE PROCESS, Introduction aux process, LIRE VARIABLE PROCESS.

41

Process (Interface utilisateur)

CACHER PROCESS (process)

Paramètre	Type		Description
process	Numérique	→	Numéro du process à cacher

Description

CACHER PROCESS masque toutes les fenêtres appartenant au process dont le numéro est process. Tous les éléments d'interface de process sont cachés jusqu'au MONTRER PROCESS suivant. La barre de menus du process est aussi cachée. L'ouverture d'une fenêtre alors que le process est caché ne provoquera aucun redessinement d'écran. Si le process est déjà caché, cette commande ne fait rien.

La seule exception à cette règle est la fenêtre du débogueur. Si la fenêtre du débogueur est affichée lorsque process est caché, process est affiché et passe au premier plan.

Si vous ne voulez pas qu'un process soit affiché lorsqu'il est créé, CACHER PROCESS doit être la première commande à appeler dans la méthode du process. Les process Process principal et Gestionnaire du cache ne peuvent pas être cachés à l'aide de cette commande.

Lorsqu'un process est caché, il est toujours en cours d'exécution.

Si vous souhaitez ne cacher qu'une fenêtre du process, utilisez la commande CACHER FENETRE.

Exemple

L'exemple suivant cachera toutes les fenêtres appartenant au process courant :

```
⇒ CACHER PROCESS (Numero du process courant)
```

Référence

MONTRER PROCESS, Statut du process, CACHER FENETRE.

MONTRER PROCESS (process)

Paramètre	Type		Description
process	Numérique	→	Numéro du process dont les fenêtres doivent être affichées

Description

MONTRER PROCESS fait apparaître l'ensemble des fenêtres appartenant à process. Cette commande ne passe pas les fenêtres de process au premier plan, utilisez pour cela la commande PASSER AU PREMIER PLAN.

Si les fenêtres de process sont déjà affichées, cette commande ne fait rien.

Exemple

L'exemple suivant affiche le process "Clients", s'il était caché auparavant. Le numéro de process est stocké dans la variable interprocess <>Clients :

```
⇒ MONTRER PROCESS (<>Clients)
```

Référence

CACHER PROCESS, PASSER AU PREMIER PLAN, Statut du process.

PASSER AU PREMIER PLAN (process)

Paramètre	Type		Description
process	Numérique	→	Numéro du process à passer au premier plan

Description

PASSER AU PREMIER PLAN passe les fenêtres du process de numéro process au premier plan. Toutes les fenêtres appartenant à process passent au premier plan. L'ordre des fenêtres est conservé. Si le process est déjà au premier plan, la commande ne fait rien. Si le process est caché, il faut utiliser la commande MONTRER PROCESS pour faire d'abord apparaître le process, sinon PASSER AU PREMIER PLAN ne fait rien.

Le Process principal et le Process de structure peuvent être passés au premier plan à l'aide de cette commande.

Exemple

L'exemple suivant est une méthode qui peut être exécutée à partir d'une commande de menu. Elle vérifie si le process au premier plan est le process <>Clients. Sinon, ce process passe au premier plan :

```
   Si (Process de premier plan#<>Clients)  ` Si la liste des clients n'est pas affichée
⇒     PASSER AU PREMIER PLAN (<>Clients)  ` Passer cette liste au premier plan
   Fin de si
```

Référence

CACHER PROCESS, MONTRER PROCESS, Statut du process.

Process de premier plan {(*)} → Entier

Paramètre	Type		Description
*		→	Numéro du process de la première fenêtre non-flottante
Résultat	Entier	←	Numéro du process dont la ou les fenêtre(s) est (sont) au premier plan

Description

Process de premier plan retourne le numéro du process dont la ou les fenêtre(s) est (sont) au premier plan.

Lorsqu'une ou plusieurs fenêtres flottantes sont ouvertes, deux niveaux différents de fenêtres sont distingués :
• les fenêtres standard
• les fenêtres flottantes

Si la fonction Process de premier plan est utilisée dans la méthode formulaire ou dans une méthode objet d'une fenêtre flottante, la fonction retourne le numéro du process de la fenêtre flottante au premier plan parmi les fenêtres flottantes. Si vous passez le paramètre optionnel astérisque, la fonction retourne le numéro du process dont la fenêtre est au premier plan, **exception faite du niveau des fenêtres flottantes**.

Exemple

Référez-vous à l'exemple de PASSER AU PREMIER PLAN.

Référence

LISTE FENETRES, PASSER AU PREMIER PLAN.

42

Propriétés des objets

Les commandes de propriétés des objets

Les commandes de propriétés des objets sont les suivantes :

- CHANGER JEU DE CARACTERES
- CHANGER TAILLE
- CHANGER STYLE
- ACTIVER BOUTON
- INACTIVER BOUTON
- TITRE BOUTON
- CHOIX ENUMERATION
- CHOIX SAISISSABLE
- CHOIX VISIBLE
- CHOIX FORMATAGE
- CHOIX FILTRE SAISIE
- CHOIX COULEUR
- FIXER COULEURS RVB
- LIRE RECT OBJET
- DEPLACER OBJET

Les commandes de propriétés des objets agissent sur les propriétés des objets présents dans les formulaires. Elles vous permettent de modifier l'apparence et le comportement de ces objets lorsque vous utilisez les formulaires dans les modes Utilisation et Menus créés.

Important : La portée (l'aire d'action) de ces commandes est le formulaire en cours d'utilisation ; les modifications ne sont pas conservées lorsque vous sortez du formulaire.

Accès aux objets par leur nom d'objet ou à partir du nom de la source de données

Les commandes de propriétés des objets partagent toutes la même syntaxe :

NOM DE LA COMMANDE({*;} objet { ; paramètres spécifiques à la commande)

Si vous spécifiez le paramètre optionnel *, vous indiquez un nom d'objet (une chaîne) dans objet.

Vous pouvez utiliser le caractère @ dans ce nom si vous voulez adresser plusieurs objets du formulaire dans un seul appel. Le tableau qui suit montre des exemples de noms d'objets que vous pouvez spécifier pour cette commande.

Noms d'objets	Objets affectés par l'appel
zoneGroupe	Uniquement l'objet zoneGroupe.
zone@	Les objets dont le nom commence par “zone”.
@zoneGroupe	Les objets dont le nom finit par “zoneGroupe”.
@Groupe@	Les objets dont le nom contient “Groupe”.
zone@Btn	Les objets dont le nom commence par “zone” et finit par “Btn”.
@	Tous les objets présents dans le formulaire.

Note : A compter de la version 6.5 de 4D, il est possible de paramétrer le mode d’interprétation du caractère @, lorsque celui-ci est inclus dans une chaîne de caractères. Cette option influe sur le fonctionnement des commandes du thème “Propriétés des objets”. Pour plus d'informations, reportez-vous au manuel *Mode Structure* de 4D.

Si vous omettez le paramètre optionnel *, vous indiquez un champ ou une variable dans objet. Dans ce cas, vous ne spécifiez pas une chaîne mais une référence de champ ou de variable (champ et variable objet uniquement).

Note : Cette seconde syntaxe reste compatible avec les versions précédentes de 4D.

Référence

ACTIVER BOUTON, CHANGER JEU DE CARACTERES, CHANGER STYLE, CHANGER TAILLE, CHOIX ENUMERATION, CHOIX FILTRE SAISIE, CHOIX FORMATAGE, CHOIX SAISISSABLE, CHOIX VISIBLE, FIXER COULEURS RVB, INACTIVER BOUTON, TITRE BOUTON.

CHANGER JEU DE CARACTERES ({*; }objet; police)

Paramètre	Type		Description
*		→	Si spécifié = objet est un nom d'objet (chaîne) Si omis = objet est un champ ou une variable
objet	Objet	→	Nom d'objet (si * spécifié) ou Champ ou Variable (si * omis)
police	Chaîne | Num	→	Nom ou numéro de police de caractères

Description

CHANGER JEU DE CARACTERES passe la police dans laquelle objet est affiché en police. Le paramètre police peut être soit un nom soit un numéro de police.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section d'introduction, Propriétés des objets.

Exemples

(1) L'exemple suivant définit la police d'un bouton nommé bOK. La police est Arial, une police système sous Windows :

```
` Modification de la police de bOK
⇒ CHANGER JEU DE CARACTERES (bOK; "Arial")
```

(2) L'exemple suivant définit la police de tous les objets d'un formulaire dont le nom contient "info".

```
⇒ CHANGER JEU DE CARACTERES (*;"@info@";"Times")
```

Référence

CHANGER STYLE, CHANGER TAILLE.

CHANGER TAILLE ({*; }objet; taille)

Paramètre	Type		Description
*		→	Si spécifié = objet est un nom d'objet (chaîne) Si omis = objet est un champ ou une variable
objet	Objet	→	Nom d'objet (si * est spécifié) ou Champ ou variable (si * est omis)
taille	Numérique	→	Taille de police en points

Description

CHANGER TAILLE définit la taille de la police du ou des objet(s) de formulaire spécifié(s) par objet.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

La taille peut être tout Entier compris entre 1 et 255. Si la taille exacte n'existe pas, les caractères sont proportionnellement redimensionnés.

La zone de l'objet, telle qu'elle a été définie dans le formulaire, doit être suffisamment grande pour afficher les données dans la nouvelle taille. Autrement, le texte peut être tronqué ou pas du tout affiché.

Exemples

(1) L'exemple suivant définit la taille de police de la variable appelée vInfo :

```
⇒ CHANGER TAILLE (vInfo; 14)
```

(2) L'exemple suivant définit la taille de police de tous les objets de formulaire dont le nom débute par "hl" :

```
⇒ CHANGER TAILLE (*;"hl@";14)
```

CHANGER STYLE ({*; }objet; style)

Paramètre	Type		Description
*		→	Si spécifié = objet est un nom d'objet (chaîne) Si omis = objet est un champ ou une variable
objet	Objet	→	Nom d'objet (si * est spécifié) ou Champ ou variable (si * est omis)
style	Numérique	→	Style de police

Description

CHANGER STYLE assigne le style de police style à ou aux objet(s) de formulaire désigné(s) par objet.
Le nombre style est un code de style du système d'exploitation. En additionnant des codes, vous combinez les styles.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

Vous devez passer dans le paramètre style une des constantes prédéfinies suivantes ou la somme de plusieurs de ces constantes :

Constante	Type	Valeur
Normal	Entier long	0
Gras	Entier long	1
Italique	Entier long	2
Souligné	Entier long	4
Relief	Entier long	8
Ombré	Entier long	16
Condensé	Entier long	32
Etendu	Entier long	64

Note : Sous Windows, seuls les styles Normal, Gras, Italique et Souligné sont disponibles.

Exemples

(1) L'exemple suivant définit le style de police du bouton bAjoutNouveau. Le style demandé est gras italique :

⇒ **CHANGER STYLE** (bAjoutNouveau; Gras + Italique)

(2) L'exemple suivant définit le style de police Normal pour tous les objets de formulaire dont le nom débute par "vt" :

⇒ **CHANGER STYLE** (*; "vt@";Normal)

Référence

CHANGER JEU DE CARACTERES, CHANGER TAILLE.

ACTIVER BOUTON ({*; }objet)

Paramètre	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)

Description

La commande ACTIVER BOUTON active le ou les objet(s) de formulaire désigné(s) par objet.

Un bouton ou un objet activé réagit aux clics souris et aux raccourcis clavier.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si ne passez pas ce paramètre, vous indiquez que le paramètre objet est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable (variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

Cette commande (malgré ce que son nom suggère) peut être appliquée aux types d'objets suivants :

- Bouton, Bouton par défaut, Bouton 3D, Bouton invisible, Bouton inversé
- Bouton radio, Bouton radio 3D, Bouton image
- Case à cocher, Case à cocher 3D
- Pop-up menu, Liste déroulante, Combo Box, Menu/Liste déroulante
- Thermomètre, Règle

Note : Il est inutile d'appeler cette commande avec un objet auquel une action automatique a été assignée car 4D se charge de modifier son état lorsque c'est nécessaire.

Exemples

(1) L'exemple suivant active le bouton bValider :

```
⇒ ACTIVER BOUTON(bValider)
```

(2) L'exemple suivant active tous les objets du formulaire dont le nom contient “btn”:

⇒ **ACTIVER BOUTON**(*;"@btn@")

(3) Reportez-vous l'exemple de la commande TITRE BOUTON.

Référence
INACTIVER BOUTON, TITRE BOUTON.

INACTIVER BOUTON ({*; }objet)

Paramètre	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)

Description

La commande INACTIVER BOUTON inactive le ou les objet(s) de formulaire désigné(s) par objet.

Un bouton ou un objet inactivé ne réagit pas aux clics souris ni aux raccourcis clavier, et est affiché en grisé.

Note : Désactiver un bouton ou un objet ne vous empêche pas de modifier sa valeur par programmation.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre objet est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable (variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

Cette commande (malgré ce que son nom suggère) peut être appliquée aux types d'objets suivants :

- Bouton, Bouton par défaut, Bouton 3D, Bouton invisible, Bouton inversé
- Bouton radio, Bouton radio 3D, Bouton image
- Case à cocher, Case à cocher 3D
- Pop-up menu, Liste déroulante, Combo Box, Menu/Liste déroulante
- Thermomètre, Règle

Note : Il est inutile d'appeler cette commande avec un objet auquel une action automatique a été assignée car 4D se charge de modifier son état lorsque c'est nécessaire.

Exemples

(1) L'exemple suivant inactive le bouton bValider :

⇒ **INACTIVER BOUTON**(bValider)

(2) L'exemple suivant inactive tous les objets de formulaire dont le nom contient “btn” :

⇒ **INACTIVER BOUTON**(*;"@btn@")

(3) Reportez-vous à l'exemple de la commande TITRE BOUTON.

Référence

ACTIVER BOUTON, TITRE BOUTON.

TITRE BOUTON ({*; }objet; libellBouton)

Paramètre	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)
libellBouton	Alpha	→	Nouveau nom du bouton

Description

La commande TITRE BOUTON change le titre du ou des bouton(s) spécifié(s) dans le paramètre objet et le remplace par la valeur définie dans le paramètre libellBouton.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre objet est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable (variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

TITRE BOUTON n'affecte que les boutons affichant du texte : boutons, cases à cocher et boutons radio.

Généralement, cette commande s'applique à un bouton à la fois. La zone du bouton doit être assez grande pour pouvoir accueillir le texte ; sinon, le texte est tronqué. N'utilisez pas de retours chariot dans libellBouton.

Exemple

L'exemple suivant est la méthode objet d'un bouton de recherche situé dans la zone de pied de page d'un formulaire sortie affiché par la commande MODIFIER SELECTION. La méthode effectue une recherche dans une table et active ou inactive le bouton intitulé bSuppr et change son titre, en fonction des résultats de la recherche :

```
CHERCHER ([Personnes]; [Personnes]Nom = vNom)
Au cas ou
   : (Enregistrements trouves ([Personnes]) = 0) ` Personne n'a été trouvé
⇒     TITRE BOUTON (bSuppr;" Supprimer")
      INACTIVER BOUTON (bSuppr)
   : (Enregistrements trouves ([Personnes]) = 1) ` Une personne a été trouvée
⇒     TITRE BOUTON (bSuppr;"Supprimer la personne")
      ACTIVER BOUTON (bSuppr)
   : (Enregistrements trouves([Personnes]) > 1) ` Plusieurs personnes ont été trouvées
⇒     TITRE BOUTON (bSuppr;"Supprimer les personnes")
      ACTIVER BOUTON (bSuppr)
Fin de cas
```

Référence

ACTIVER BOUTON, INACTIVER BOUTON.

CHOIX FORMATAGE ({*; }objet; formatAffich)

Paramètre	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est un champ ou une variable
objet	Objet de formulaire	→	Nom d'objet (si * spécifié) ou Champ ou Variable (si * omis)
formatAffich	Alpha	→	Nouveau format d'affichage de l'objet

Description

CHOIX FORMATAGE remplace le format d'affichage du ou des objet(s) spécifié(s) par objet avec le format que vous avez passé dans formatAffich.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

La commande CHOIX FORMATAGE peut être indifféremment utilisée dans des formulaires entrée ou sortie (affichés ou imprimés) et peut être appliquée aux champs, variables saisissables/non saisissables et zones de défilement numériques.

Vous devez utiliser un format d'affichage compatible avec le type de données présentes dans l'objet :

• Pour formater des champs booléens, passez deux valeurs séparées par un point-virgule (;).

• Pour formater des champs ou variables de type Date, passez Caractere(n) dans formatAffich, où n peut être une des constantes prédéfinies suivantes :

Constante	Type	Valeur
Format court	Entier long	1
Format abrégé	Entier long	2
Format long	Entier long	3
Format spécial	Entier long	4
Jour Mois Année	Entier long	5
Abrégé Jour Mois Année	Entier long	6
Spécial forcé	Entier long	7

• Pour formater des champs ou variables de type Heure, passez Caractere(n) dans formatAffich, où n peut être une des constantes prédéfinies suivantes :

Constante	Type	Valeur
h mn s	Entier long	1
h mn	Entier long	2
Heure Minute Seconde	Entier long	3
Heure Minute	Entier long	4
h mn Matin Après Midi	Entier long	5

• Pour formater des champs ou variables de type Image, passez Caractere(n) dans formatAffich, où n peut être une des constantes prédéfinies suivantes :

Constante	Type	Valeur
Tronquée centrée	Entier long	1
Non tronquée	Entier long	2
Sur fond	Entier long	3
Tronquée non centrée	Entier long	4
Proportionnelle	Entier long	5
Proportionnelle centrée	Entier long	6

Pour plus d'informations sur les formats d'affichage alphanumériques et numériques, reportez-vous au manuel *Mode Structure* de 4e Dimension.

Note : Pour pouvoir exploiter les formats d'affichage que vous avez créés dans la boîte de dialogue des Propriétés de la base, préfixez le nom du format, dans le paramètre formatAffich, d'une barre verticale (|).

Exemples

(1) La ligne de code suivante formate le champ [Employés]Date embauche au cinquième format de date.

⇒ **CHOIX FORMATAGE** ([Employés]Date embauche; **Caractere**(Jour Mois Année))

(2) L'exemple suivant change le format d'un champ [Sociétés]Code postal selon la longueur du code postal :

```
    Si (Longueur ([Sociétés]Code postal) = 9)
⇒       CHOIX FORMATAGE ([Sociétés]Code postal; "#####-####")
    Sinon
⇒       CHOIX FORMATAGE ([Sociétés]Code postal; "#####")
    Fin de si
```

(3) L'exemple suivant définit le format d'un champ booléen pour afficher soit "Marié" soit "Célibataire" au lieu des valeurs par défaut "Oui" et "Non" :

⇒ **CHOIX FORMATAGE** ([Employés]Situation; "Marié;Célibataire")

Référence

CHOIX FILTRE SAISIE.

CHOIX FILTRE SAISIE ({*; }objet; filtreSaisie)

Paramètre	Type		Description
*		→	Si spécifié = objet est un nom d'objet (chaîne) Si omis = objet est un champ ou une variable
objet	Objet	→	Nom d'objet (si * spécifié) ou Champ ou Variable (si * omis)
filtreSaisie	Alpha	→	Nouveau filtre de saisie pour la zone saisissable

Description

CHOIX FILTRE SAISIE remplace le filtre de saisie pour objet par filtreSaisie dans le formulaire courant affiché à l'écran.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

La commande CHOIX FILTRE SAISIE peut être utilisée dans des formulaires entrée et des dialogues et peut être appliquée aux champs et variables saisissables acceptant les filtres de saisie en mode Structure.

Pour enlever un filtre, passez une chaîne vide dans le paramètre filtreSaisie.

Note : Cette commande ne peut pas être utilisée avec des champs situés dans le formulaire "liste" d'un sous-formulaire.

Note : Pour pouvoir exploiter les filtres de saisie que vous avez créés dans la boîte de dialogue des Propriétés de la base, préfixez le nom du filtre, dans le paramètre filtreSaisie, d'une barre verticale (|).

Exemples

(1) L'exemple suivant définit le filtre de saisie pour le champ code postal. Si l'adresse se trouve en France, le filtre est paramétré pour les codes postaux français. Sinon, le filtre peut accepter toute valeur saisie :

```
      Si (Pays = "France")  ` Fixer le filtre au format du code postal français
⇒        CHOIX FILTRE SAISIE ([Sociétés]Code postal; &"#####")
      Sinon  ` Fixer le filtre pour qu'il accepte toute valeur alphanumérique
⇒        CHOIX FILTRE SAISIE ([Sociétés]Code postal; "~@")
      Fin de si
```

(2) L'exemple suivant autorise uniquement la saisie des lettres “a”, “b”, “c” ou “g” dans un champ comportant deux lettres :

```
⇒     CHOIX FILTRE SAISIE([Table]Champ;" &"+Caractere (Guillemets doubles)+ "a;b;c;g" +
                                          Caractere (Guillemets doubles)+"##")
```

Note : Cet exemple définit le filtre de saisie &"a;b;c;g"##.

Référence

CHOIX FORMATAGE.

CHOIX ENUMERATION ({*; }objet; énum)

Paramètre	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est un champ ou une variable
objet	Objet de formulaire	→	Nom d'objet (si * spécifié) ou Champ ou Variable (si * omis)
énum	Alpha	→	Nom de la liste à utiliser comme énumération (telle que définie en mode Structure)

Description

La commande CHOIX ENUMERATION construit ou remplace l'énumération du ou des objet(s) spécifié(s) par objet avec la liste hiérarchique créée dans l'éditeur d'énumérations, en mode Structure, et passée dans énum.

Cette commande peut être appliquée, dans un formulaire entrée ou un dialogue, aux champs et variables saisissables dont les valeurs peuvent être saisies sous forme de texte. La liste s'affiche pendant la saisie lorsque l'utilisateur sélectionne la zone de texte.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

Note : Cette commande ne peut pas être utilisée avec des champs placés dans le formulaire "liste" d'un sous-formulaire.

Exemple

L'exemple suivant définit l'énumération liée à un champ Coursiers. Si l'envoi doit être effectué de nuit, alors l'énumération affiche les sociétés de courses qui fonctionnent la nuit. Sinon, l'énumération standard est proposée :

```
      Si ([Courses]Nuit)
⇒        CHOIX ENUMERATION([Courses]Coursier; "Coursiers de nuit")
      Sinon
⇒        CHOIX ENUMERATION([Courses]Coursier; "Coursiers standard")
      Fin de si
```

CHOIX SAISISSABLE ({*; }objet; zoneSaisie)

Paramètre	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est un champ ou une variable
objet	Objet de formulaire	→	Nom d'objet (si * spécifié) ou Champ ou Variable (si * omis)
zoneSaisie	Booléen	→	Vrai = saisissable ; Faux = non saisissable

Description

CHOIX SAISISSABLE rend saisissable ou non saisissable le ou les objet(s) de formulaire désigné(s) par objet.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

L'utilisation de cette commande est équivalente à la sélection de l'option saisissable ou non saisissable pour un champ ou une variable dans la fenêtre des Propriétés d'objet de l'éditeur de formulaires. CHOIX SAISISSABLE fonctionne avec un sous-formulaire uniquement si elle se trouve dans la méthode formulaire du sous-formulaire.

Lorsque zoneSaisie est saisissable (Vrai), l'utilisateur peut y placer le curseur pour saisir des données. Lorsque zoneSaisie est non saisissable (Faux), l'utilisateur ne peut pas placer le curseur dans la zone et ne peut donc pas saisir de valeurs.

Rendre un objet non saisissable n'empêche pas sa modification par programmation.

Exemple

L'exemple suivant définit un champ de type d'expédition suivant le poids d'un colis expédié. Si le colis pèse un kilo ou moins, l'expéditeur sera La Poste et le champ est rendu non saisissable. Sinon, le champ est rendu saisissable.

```
Si ([Expédition]Poids <= 1)
    [Expédition]Type := "La Poste"
⇒   CHOIX SAISISSABLE ([Expédition]Type; Faux)
Sinon
    CHOIX SAISISSABLE ([Expédition]Type; Vrai)
Fin de si
```

Référence

ACTIVER BOUTON, CHOIX VISIBLE, INACTIVER BOUTON.

CHOIX VISIBLE ({*; }objet; visible)

Paramètre	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est un champ ou une variable
objet	Objet de formulaire	→	Nom d'objet (si * est passé) ou Champ ou Variable (si * est omis)
visible	Booléen	→	Vrai = visible, Faux = invisible

Description

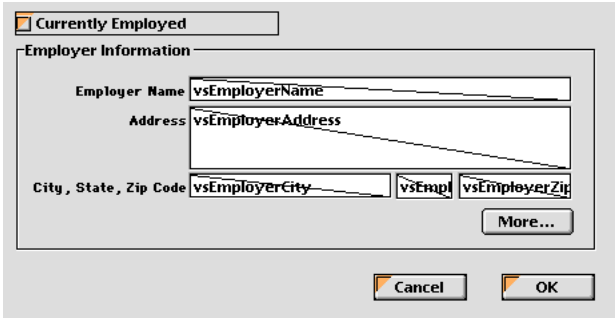
La commande CHOIX VISIBLE affiche ou masque le ou les objet(s) défini(s) par les paramètres objet et *.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet désigne le nom d'un objet (une chaîne). Si vous ne passez pas le paramètre *, vous indiquez que le paramètre objet désigne un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne de caractères mais la référence du champ ou de la variable (champs ou variables objets uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

Si vous passez la valeur VRAI dans le paramètre visible, le ou les objet(s) sont affichés. Si vous passez FAUX dans visible, les objets sont masqués.

Exemple

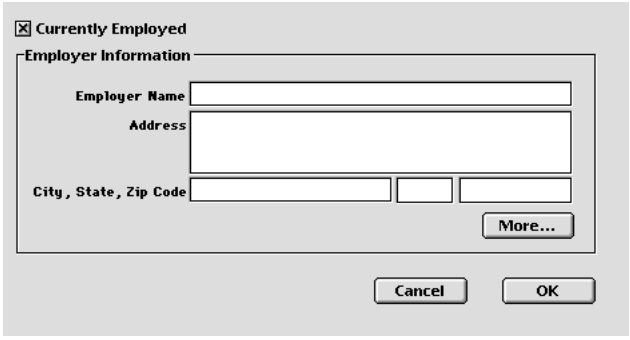
Voici un formulaire tel qu'il apparaît en mode Structure :



Les objets dans la zone de groupe **Employer Information** ont tous un nom qui contient l'expression “employer” (y compris la zone de groupe). Lorsque l'option **Currently Employed** est cochée, les objets doivent être visibles, lorsqu'elle est désélectionnée les objets doivent être invisibles. Voici la méthode objet de la case à cocher :

```
      ` Méthode objet Case à cocher cbCurrentlyEmployed
   Au cas ou
      : (Evenement formulaire=Sur chargement)
         cbCurrentlyEmployed:=1
      : (Evenement formulaire=Sur clic)
            ` Cacher ou montrer tous les objets dont le nom contient "emp"
⇒        CHOIX VISIBLE(*;"@emp@";cbCurrentlyEmployed # 0)
            ` Mais conserver la case à cocher toujours visible
⇒        CHOIX VISIBLE(cbCurrentlyEmployed;Vrai)
   Fin de cas
```

En mode Utilisation ou Menus créés, le formulaire apparaîtra ainsi :



ou ainsi :

CHOIX COULEUR ({*; }objet; couleur)

Paramètre	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est un champ ou une variable
objet	Champ|Variable	→	Nom d'objet (si * spécifié) ou Champ ou Variable (si * omis)
couleur	Numérique	→	Nouvelles couleurs pour l'objet

Description

La commande CHOIX COULEUR définit les couleurs de premier plan et d'arrière-plan du ou des objet(s) de formulaire spécifié(s) par objet.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne). Si ne passez pas le paramètre, vous indiquez que le paramètre objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

Le paramètre couleur définit à la fois les couleurs de premier plan et d'arrière-plan. La couleur est calculée de la manière suivante : Couleur:=–(Premier_Plan+(256 * Arrière_Plan)), où Premier_Plan et Arrière_Plan sont des numéros de couleur (de 0 à 255) parmi la palette de couleurs de 4D — que vous pouvez visualiser, par exemple, dans la fenêtre des propriétés d'objets de l'éditeur de formulaires. Couleur est toujours un nombre négatif. Par exemple, si la couleur de premier plan est 20 et si la couleur d'arrière-plan est 10, alors couleur est égal à – (20 + (256 * 10)) soit –2580.

Les numéros les plus souvent utilisés sont fournis par 4e Dimension sous forme de constantes prédéfinies :

Constante	Type	Valeur
Blanc	Entier long	0
Jaune	Entier long	1
Orange	Entier long	2
Rouge	Entier long	3
Violet	Entier long	4
Bleu foncé	Entier long	5
Bleu	Entier long	6
Bleu clair	Entier long	7
Vert	Entier long	8
Vert foncé	Entier long	9
Marron foncé	Entier long	10
Gris foncé	Entier long	11
Gris clair	Entier long	12
Marron	Entier long	13
Gris	Entier long	14
Noir	Entier long	15

Note : Tandis que CHOIX COULEUR travaille avec des couleurs indexées dans la palette de couleurs de 4D, il est à noter que la version 6 a introduit la nouvelle commande FIXER COULEURS RVB qui vous permet de travailler avec toute couleur RVB.

Exemple

L'exemple suivant définit la couleur d'un bouton nommé bInfo. La couleur est déterminée par les deux variables vForeground et vBackground :

```
⇒ CHOIX COULEUR (bInfo; – (vForeground + (256 * vBackground)))
```

Référence

FIXER COULEURS RVB.

FIXER COULEURS RVB ({*; }objet; couleurAvantPlan; couleurArrièrePlan)

Paramètre	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est un champ ou une variable
objet	Objet	→	Nom d'objet (si * est spécifié) ou Champ ou Variable (si * est omis)
couleurAvantPlan	Numérique	→	Valeur de la couleur RVB d'avant-plan
couleurArrièrePlan	Numérique	→	Valeur de la couleur RVB d'arrière-plan

Description

La commande FIXER COULEURS RVB modifie les couleurs d'avant-plan et d'arrière-plan du ou des objet(s) défini(s) par le paramètre objet et le paramètre optionnel *.

Si vous passez le paramètre optionnel *, vous spécifiez que le paramètre objet est le nom d'un objet (une chaîne de caractères). Si le paramètre * est omis, vous spécifiez que objet est un champ ou un objet. Dans ce cas, vous ne passez pas dans objet une chaîne de caractères mais la référence à un champ ou à une variable (champ ou variable objet uniquement). Pour plus d'informations sur les noms d'objets, reportez-vous à la section Propriétés des objets.

Vous passez des valeurs de couleurs RVB dans les paramètres couleurAvantPlan et couleurArrièrePlan. Ces valeurs sont des entiers longs de 4 octets dont le format (0x00RRGGBB) est décrit ci-dessous (les octets sont numérotés de 0 à 3 de la droite vers la gauche) :

Octet	Description
3	Doit être zéro pour une couleur RVB absolue
2	Composante rouge de la couleur (0..255)
1	Composante verte de la couleur (0..255)
0	Composante bleue de la couleur (0..255)

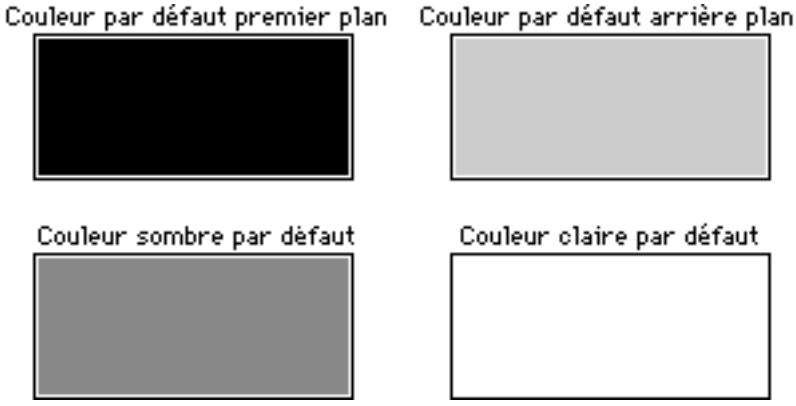
Le tableau ci-dessous présente des exemples de valeurs de couleurs RVB :

Valeur	Description
0x00000000	Noir
0x00FF0000	Rouge vif
0x0000FF00	Vert vif
0x000000FF	Bleu vif
0x007F7F7F	Gris
0x00FFFF00	Jaune vif
0x00FF7F7F	Rouge pastel
0x00FFFFFF	Blanc

Vous pouvez aussi spécifier une des quatre couleurs utilisées par défaut par 4e Dimension pour dessiner des objets ayant la propriété de couleur "automatique". Les constantes prédéfinies suivantes sont proposées par 4e Dimension :

Constante	Type	Valeur
Couleur par défaut premier plan	Entier long	-1
Couleur par défaut arrière plan	Entier long	-2
Couleur sombre par défaut	Entier long	-3
Couleur claire par défaut	Entier long	-4

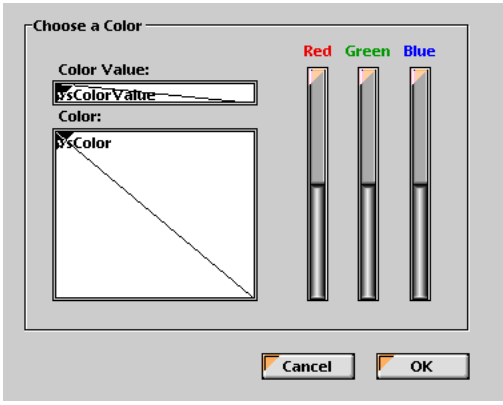
Ces couleurs (sur un système standard) sont les suivantes :



ATTENTION : Notez que, sous Windows, ces couleurs automatiques dépendent du système. Si vous modifiez vos couleurs système dans le Panneau de configuration "Affichage", les couleurs automatiques de 4e Dimension seront modifiées en conséquence. Utilisez les valeurs de couleurs automatiques pour assigner à des objets les couleurs système, et non pour leur assigner les mêmes couleurs que celles définies dans l'exemple ci-dessus.

Exemple

Voici un formulaire contenant deux variables non saisissables, vsColorValue et vsColor ainsi que trois thermomètres, thRouge, thVert et thBleu :



Les méthodes associées à ces objets sont les suivantes :

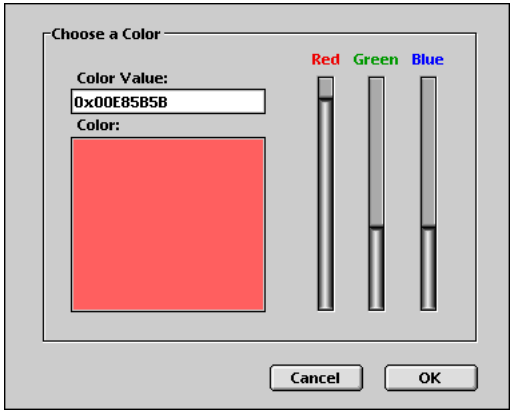
```
   ` Méthode objet de la variable non saisissable vsColorValue
Au cas ou
   : (Evenement formulaire=Sur chargement)
      vsColorValue:="0x00000000"
Fin de cas

   ` Méthode objet de la variable non saisissable vsColor
Au cas ou
   : (Evenement formulaire=Sur chargement)
      vsColor:=""
⇒     FIXER COULEURS RVB(vsColor;0x00FFFFFF;0x0000)
Fin de cas

   ` Méthode objet du thermomètre thRouge
CLIC SUR THERMOMETRE COULEUR

   ` Méthode objet du thermomètre thVert
CLIC SUR THERMOMETRE COULEUR

   ` Méthode objet du thermomètre thBleu
CLIC SUR THERMOMETRE COULEUR
```

La méthode projet appelée par les trois thermomètres est la suivante :

```
      ` Méthode projet CLIC SUR THERMOMETRE COULEUR
⇒     FIXER COULEURS RVB(vsColor;0x00FFFFFF;(thRouge << 16)+(thVert << 8)+thBleu)
      vsColorValue:=Chaine((thRouge << 16)+(thVert << 8)+thBleu;"&x")
      Si (thRouge=0)
         vsColorValue:=Sous chaine(vsColorValue;1;2)+"0000"+Sous chaine(vsColorValue;3)
      Fin de si
```

Notez l'utilisation des Opérateurs sur les bits pour le calcul des valeurs des couleurs à partir de celles des thermomètres.

Dans les modes Utilisation ou Menus créés, le formulaire a l'aspect suivant :



Référence

CHOIX COULEUR, Opérateurs sur les bits.

LIRE RECT OBJET ({*; }objet; gauche; haut; droite; bas)

Paramètre	Type		Description
*	*	→	Si spécifié = objet est un nom d'objet (chaîne) Si omis = objet est une variable
objet	Objet	→	Nom d'objet (si * est spécifié) ou Champ ou variable (si * est omis)
gauche	Entier long	←	Coordonnée gauche de l'objet
haut	Entier long	←	Coordonnée supérieure de l'objet
droite	Entier long	←	Coordonnée droite de l'objet
bas	Entier long	←	Coordonnée inférieure de l'objet

Description

La commande LIRE RECT OBJET retourne dans les variables ou champs gauche, haut, droite et bas les coordonnées (en points) du ou des objet(s) du formulaire courant défini(s) par les paramètres * et objet.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne de caractères). Si vous ne passez pas le paramètre *, vous indiquez que objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable de type objet uniquement).

Si vous passez un nom d'objet dans le paramètre objet et utilisez le caractère joker @ afin de sélectionner plusieurs objets, les coordonnées retournées seront celles du rectangle formé par l'ensemble des objets concernés.

Note : Il est possible de paramétrer le mode d'interprétation du caractère @, lorsque celui-ci est inclus dans une chaîne de caractères. Cette option influe sur le fonctionnement des commandes du thème "Propriétés des objets". Pour plus d'informations, reportez-vous au manuel *Mode Structure*.

Si l'objet n'existe pas ou si la commande est appelée ailleurs que dans le contexte d'un formulaire, les coordonnées retournées sont (0;0;0;0).

Exemple

Vous souhaitez obtenir les coordonnées du rectangle formé par tous les objets dont le nom commence par "bouton" :

```
⇒ LIRE RECT OBJET(*;"bouton@";gauche;haut;droite;bas)
```

Référence

DEPLACER OBJET.

DEPLACER OBJET ({*; }objet; dépH; dépV{; redimH{; redimV{; *}}})

Paramètre	Type		Description
*	*	→	Si spécifié = objet est un nom d'objet (chaîne) Si omis = objet est une variable
objet	Objet	→	Nom d'objet (si * est spécifié) ou Champ ou variable (si * est omis)
dépH	Entier long	→	Valeur de déplacement horizontal de l'objet (>0 = vers la droite, <0 = vers la gauche)
dépV	Entier long	→	Valeur de déplacement vertical de l'objet (>0 = vers le bas, <0 = vers le haut)
redimH	Entier long	→	Valeur de redimensionnement horizontal de l'objet
redimV	Entier long	→	Valeur de redimensionnement vertical de l'objet
*	*	→	Si spécifié = coordonnées absolues Si omis = coordonnées relatives

Description

La commande DEPLACER OBJET permet de déplacer le ou les objet(s) du formulaire courant, défini(s) par les paramètres * et objet, de dépH pixels horizontalement et de dépV pixels verticalement.
Il est également possible (optionnellement) de redimensionner le ou les objet(s) de redimH pixels horizontalement et de redimV pixels verticalement.

Le sens du déplacement ou du redimensionnement est défini par le signe des valeurs passées dans les paramètres dépH et dépV :

• Si la valeur est positive, le déplacement ou le redimensionnement s'effectue respectivement vers la droite ou vers le bas.

• Si la valeur est négative, le déplacement ou le redimensionnement s'effectue respectivement vers la gauche ou vers le haut.

Si vous passez le premier paramètre optionnel *, vous indiquez que le paramètre objet est un nom d'objet (une chaîne de caractères). Si vous ne passez pas le paramètre *, vous indiquez que objet est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable de type objet uniquement).
Si vous passez un nom d'objet dans le paramètre objet et utilisez le caractère joker @ afin de sélectionner plusieurs objets, tous les objets sélectionnés seront déplacés ou redimensionnés.

Par défaut, les valeurs de dépH, dépV, redimH et redimV modifient les coordonnées de l'objet relativement à sa position précédente. Si vous souhaitez que ces paramètres définissent des coordonnées absolues, passez le dernier paramètre optionnel *.

Cette commande fonctionne uniquement dans les contextes suivants :

• Formulaires entrée en mode saisie,

• En-têtes et pieds de page des formulaires sortie affichés par les commandes MODIFIER SELECTION ou VISUALISER SELECTION,

• Formulaires en cours d'impression.

Exemples

(1) L'instruction suivante déplace le bouton "Bouton_1" de 10 pixels vers la droite et de 20 pixels vers le haut, et agrandit le bouton de 30 pixels en largeur et de 40 en hauteur :

⇒ **DEPLACER OBJET** (*;"Bouton_1";10;-20;30;40)

(2) L'instruction suivante place le bouton "Bouton_1" aux coordonnées (10;20) (30;40) :

⇒ **DEPLACER OBJET** (*;"Bouton_1";10;20;30;40;*)

Référence

LIRE RECT OBJET.

43

Protocole sécurisé

GENERER CLES CRYPTAGE

Protocole sécurisé

version 6.7

GENERER CLES CRYPTAGE (cléPrivée; cléPublique{; longueur})

Paramètre	Type		Description
cléPrivée	BLOB	←	BLOB devant recevoir la clé privée
cléPublique	BLOB	←	BLOB devant recevoir la clé publique
longueur	Entier long	→	Longueur des clés en bits [386...1024] Par défaut = 512

Description

La commande GENERER CLES CRYPTAGE génère une nouvelle paire de clés RSA. Ces clés, destinées au cryptage/décryptage des données, sont à la base du système de sécurisation des données proposé par 4D. Elles pourront être utilisées, dans le cadre du protocole SSL, avec le serveur Web 4D (cryptage et sécurité des communications) mais également dans toute base de données (cryptage de données).

Après l'exécution de la commande, les BLOB passés dans les paramètres cléPrivée et cléPublique contiennent une nouvelle paire de clés de cryptage.

Le paramètre optionnel longueur vous permet de préciser la taille (en bits) des clés que vous souhaitez obtenir. Plus une clé est longue, plus son décryptage "frauduleux" sera difficile.
En contrepartie, plus les clés sont longues, plus les délais d'exécution ou de réponse seront importants, en particulier dans le cadre d'une connexion SSL.
Par défaut (si vous omettez le paramètre longueur), la taille des clés générée est de 512 bits, ce qui correspond au compromis sécurité/ rapidité généralement admis. Pour augmenter la sécurité dans ce cas, vous pouvez envisager de changer de paire de clés assez fréquemment, par exemple tous les six mois.
Vous pouvez générer des clés de 1024 bits, ce qui renforce la sécurité du cryptage, mais ralentira les connexions de votre application Web.

Notes :

• Attention si vous générez des clés dans le but d'établir une demande de certificat SSL : seules les clés de 512 et de 1024 bits sont admises.
• Une grande partie des navigateurs Web actuels n'acceptent pas les clés d'une longueur supérieure à 512 bits. En outre, la version "Export" de la bibliothèque système de cryptage fournie par défaut par 4D SA ne permet pas d'exploiter des clés pivées d'une longueur supérieure à 512 bits. Pour plus d'informations sur ce point, reportez-vous à la section Services Web, Utiliser le protocole SSL.

Les clés générées par cette commande sont au format PEM (*Privacy Enhanced Mail*), ce qui signifie que leur contenu peut être copié et collé dans un eMail sans risque d'altération. Une fois que vous avez obtenu une paire de clés, vous pouvez générer un document texte (par exemple à l'aide de la commande BLOB VERS DOCUMENT) et stocker les clés dans un endroit sûr.

Important : La clé privée ne doit jamais être diffusée, sous quelque forme que ce soit.

RSA, clés privées et clés publiques

L'algorithme de cryptage RSA employé par la commande GENERER CLES CRYPTAGE est basé sur un système de cryptage à double clé : une clé privée et une clé publique. Comme son nom l'indique, la clé publique peut être diffusée auprès de tiers, et permet le décryptage des informations. Il lui correspond une clé privée unique, utilisée pour crypter les données. La clé privée sert au cryptage ; la clé publique, au décryptage (ou inversement). Ce qui est crypté avec une clé ne peut être décrypté qu'avec l'autre.
Les fonctions de cryptage du protocole SSL sont basées sur ce principe, la clé publique étant incluse dans le certificat envoyé aux browsers (cf. section Services Web, Utiliser le protocole SSL).

Ce mode de cryptage est également utilisé par la première syntaxe des commande CRYPTER BLOB et DECRYPTER BLOB. Ce principe requiert que la clé publique soit diffusée de manière confidentielle.
Il est possible de mêler les clés publiques et privées de deux intervenants pour crypter des données de telle manière que seul le récepteur peut décrypter les données, et seul l'émetteur peut les avoir cryptées. C'est le principe de la seconde syntaxe des commandes CRYPTER BLOB et DECRYPTER BLOB.

Exemple

Reportez-vous à l'exemple de la commande CRYPTER BLOB.

Référence

CRYPTER BLOB, DECRYPTER BLOB, GENERER DEMANDE CERTIFICAT.

GENERER DEMANDE CERTIFICAT

Protocole sécurisé

version 6.7

GENERER DEMANDE CERTIFICAT (cléPrivée; demCertif; tabCodes; tabLibellés)

Paramètre	Type		Description
cléPrivée	BLOB	→	BLOB contenant la clé privée
demCertif	BLOB	←	BLOB devant recevoir la demande de certificat
tabCodes	Tab Entier long	→	Liste des codes d'informations
tabLibellés	Tab Alpha	→	Liste des libellés d'informations

Description

La commande GENERER DEMANDE CERTIFICAT permet de générer une demande de certificat au format PEM (*Privacy Enhanced Mail*), directement exploitable par des autorités de certification telles que Verisign® ou Thawthe®. Le certificat est une pièce essentielle du fonctionnement du protocole SSL dans le cadre d'un serveur Web. Il est envoyé à chaque browser se connectant en mode SSL. Il contient la "carte d'identité" du site Web (reprenant les informations que vous saisissez dans la commande), ainsi que sa clé publique — permettant aux browsers de décrypter les informations reçues. En outre, le certificat contient diverses informations ajoutées par l'autorité de certification.

Note : Pour plus d'informations sur le fonctionnement du protocole SSL avec le serveur Web 4D, reportez-vous à la section Services Web, Utiliser le protocole SSL.

La demande de certificat nécessite une paire de clés générée à l'aide de la commande GENERER CLES CRYPTAGE et contient diverses informations. C'est en combinant cette demande avec d'autres paramètres qui lui sont propres, que l'autorité de certification sera en mesure de générer un certificat.

Passez dans cléPrivée un BLOB contenant la clé privée générée avec la commande GENERER CLES CRYPTAGE.

Passez dans demCertif un BLOB vide. Après l'exécution de la commande, il contiendra la demande de certificat au format PEM. Vous pouvez placer cette demande dans un fichier texte, par exemple à l'aide de la commande BLOB VERS DOCUMENT, pour la faire parvenir à l'autorité de certification.

Important : La clé privée est utilisée pour générer la demande de certificat mais ne doit pas être envoyée à l'autorité de certification.

Vous devez remplir les tableaux tabCodes (de type entier long) et tabLibellés (de type alpha) avec, respectivement, les numéros de code et les libellés des informations destinées à l'autorité de certification.

Les codes et les libellés attendus peuvent varier en fonction de l'autorité de certification et du mode d'utilisation du certificat. Toutefois, dans le cadre d'une utilisation standard du certificat (connexions d'un serveur Web via SSL), les tableaux doivent contenir les éléments suivants :

Informations à fournir	**tabCodes**	**tabLibellés** (Exemples)
CommonName : Nom du domaine	13	www.4D.fr
CountryName : Code du pays (deux lettres)	14	FR
LocalityName : Ville	15	Clichy
StateOrProvinceName : Département, Etat...	16	Hauts de Seine
OrganizationName : Raison sociale	17	4D
OrganizationUnit : Service/Personne en charge du serveur	18	Web Administrator

L'ordre dans lequel les codes et les informations sont insérés dans les tableaux n'a pas d'importance, en revanche les deux tableaux doivent être "synchronisés" : si l'élément {3} du tableau tabCodes contient la valeur *15* (nom de la ville), l'élément {3} du tableau tabLibellés doit contenir cette information, dans notre exemple *Clichy*.

Exemple

Un formulaire "Demande de certificat" comporte les six champs nécessaires à l'établissement d'une demande de certificat standard. Le bouton **Générer** crée un document sur disque contenant la demande de certificat. Le document "Cléprivée.txt" contient la clé privée (générée à l'aide la commande GENERER CLES CRYPTAGE) doit déjà être présent sur le disque.

```
   ` Méthode objet du bouton bGénérer
C_BLOB($vbcléPrivée;$vbDemandeCert)
C_ENTIER LONG($NumTable)
TABLEAU ENTIER LONG ($tLCodes;6)
TABLEAU ALPHA (80;$tAInfos;6)

$NumTable:=Table(Table du formulaire courant)
Boucle ($i;1;6)
   $tAInfos{$i}:= Champ($NumTable;$i)
   $tLCodes{$i}:=$i+12
Fin de boucle
Si (Chercher dans tableau($tAInfos;"") # -1)
   ALERTE ("Vous devez remplir tous les champs.")
Sinon
   ALERTE ("Sélectionnez votre clé privée.")
   $vhRefDoc:=Ouvrir document("")
   Si (OK=1)
      FERMER DOCUMENT($vhRefDoc)
      DOCUMENT VERS BLOB(Document;$vbcléPrivée)
   Fin de si
⇒  GENERER DEMANDE CERTIFICAT($vbcléPrivée;$vbDemandeCert;$tLCodes;
                                                              $tAInfos)
   BLOB VERS DOCUMENT ("Demande.txt";$vbDemandeCert)
Fin de si
```

Référence

GENERER CLES CRYPTAGE.

44

Recherches et tris

CHERCHER PAR EXEMPLE ({table}{; }{*})

Paramètre	Type		Description
table	Table	→	Table de laquelle une sélection d'enregistrements doit être retournée ou Table par défaut si omis
*	*	→	Masquer les barres de défilement

Description

La commande CHERCHER PAR EXEMPLE effectue la même action que la commande de menu **Recherche par formulaire**... en mode Utilisation. Cette commande affiche le formulaire entrée courant comme fenêtre de recherche. CHERCHER PAR EXEMPLE cherche dans table les données que l'utilisateur a saisies dans cette fenêtre. Le formulaire doit contenir les champs sur lesquels vous voulez que l'utilisateur puisse effectuer la recherche. La recherche est optimisée : les champs indexés sont automatiquement utilisés.
Si vous passez le paramètre optionnel *, les barres de défilement du formulaire sont masquées.

Reportez-vous au manuel *Mode Utilisation* de 4e Dimension pour plus d'informations sur l'utilisation de la commande de menu **Recherche par formulaire...** du mode Utilisation.

Exemple

La méthode dans l'exemple suivant affiche le formulaire maRecherche. Si l'utilisateur valide le formulaire et exécute la recherche (c'est-à-dire si la variable système OK prend la valeur 1), les enregistrements trouvés sont affichés :

```
     FORMULAIRE ENTREE ([Personnes]; "maRecherche")  ` Définir le formulaire entrée
⇒    CHERCHER PAR EXEMPLE ([Personnes])  ` Afficher le formulaire pour la recherche
     Si (OK = 1)  ` Si l'utilisateur valide la recherche
        VISUALISER SELECTION ([Personnes])  ` Visualiser les enregistrements trouvés
     Fin de si
```

Référence

CHERCHER, TRIER.

Variables et ensembles système

Si l'utilisateur clique sur le bouton Valider ou appuie sur Entrée, la variable système OK prend la valeur 1 et la recherche est effectuée. Si l'utilisateur clique sur Annuler ou utilise la touche d'annulation, la variable système OK prend la valeur 0 et la recherche est annulée.

CHERCHER ({table}{; critèreRecherche}{; *})

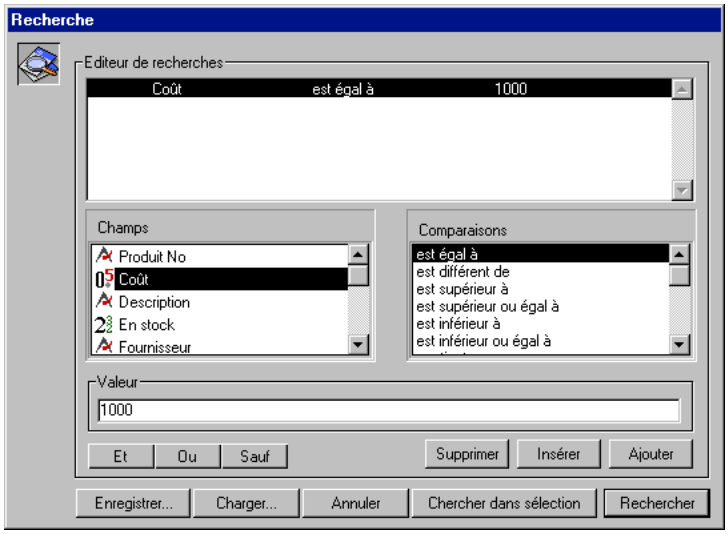
Paramètre	Type		Description
table	Table	→	Table dans laquelle la sélection est créée ou Table par défaut si ce paramètre est omis
critèreRecherche		→	Critère de recherche
*		→	Attente d'exécution de la recherche

Description

La commande CHERCHER recherche les enregistrements répondant au(x) critère(s) de recherche spécifié(s) dans critèreRecherche et retourne une sélection d'enregistrements de table. CHERCHER modifie la sélection courante de table pour le process courant. Le premier enregistrement de la nouvelle sélection devient l'enregistrement courant.

Si vous omettez le paramètre table, la commande s'applique à la table par défaut. Si aucune table par défaut n'a été définie, une erreur est générée.

Si vous ne passez ni le paramètre critèreRecherche ni le paramètre *, CHERCHER affiche la boîte de dialogue de l'Editeur de recherches de 4D pour table. Cet éditeur du mode Utilisation est présenté ci-dessous :



Pour plus d'informations sur l'utilisation de cet éditeur, reportez-vous au manuel *Mode Utilisation* de 4D.
L'utilisateur construit la recherche puis clique sur le bouton **Rechercher** ou **Chercher dans sélection**. Si la recherche est correctement effectuée et n'est pas interrompue, la variable système OK prend la valeur 1. Si l'utilisateur clique sur **Annuler**, la commande CHERCHER est interrompue sans effectuer de recherche et la variable OK prend la valeur 0 (zéro).

Exemples

(1) L'exemple suivant affiche l'Editeur de recherches pour la table [Produits] :

```
⇒ CHERCHER([Produits])
```

(2) L'exemple suivant affiche l'Editeur de recherches pour la table par défaut (si elle a été définie) :

```
⇒ CHERCHER
```

Si vous spécifiez le paramètre critèreRecherche, l'Editeur de recherches ne s'affiche pas et la recherche est entièrement définie par programmation. Pour des recherches simples (recherches sur un seul champ), vous appelez CHERCHER une seule fois avec le paramètre critèreRecherche construit de la manière décrite plus bas. Pour des recherches complexes (recherches sur de multiples champs ou avec de multiples conditions), vous appelez CHERCHER autant de fois que nécessaire avec le paramètre critèreRecherche et le paramètre optionnel * sauf pour la dernière ligne CHERCHER (qui déclenche la recherche).

(3) L'exemple suivant recherche les [Personnes] dont le nom commence par "a" :

```
⇒ CHERCHER([Personnes];[Personnes]Nom="a@")
```

(4) L'exemple suivant recherche les [Personnes] dont le nom commence par "a" ou "b" :

```
   `* indique qu'il y a un autre critère de recherche
⇒ CHERCHER([Personnes];[Personnes]Nom="a@";*)
   ` Pas de * : cela indique la fin de la définition des critères et lance l'exécution de
   ` la recherche
⇒ CHERCHER([Personnes]; | [Personnes]Nom="b@")
```

Construction d'une ligne de recherche

• Le paramètre critèreRecherche utilise la syntaxe suivante :

{opérateur ; } champ comparateur valeur

• L'opérateur est utilisé pour lier deux appels à CHERCHER lors d'une définition de recherche complexe. Les opérateurs disponibles sont les mêmes que ceux proposés dans l'Editeur de recherches en mode Utilisation :

Opérateur	Symbole
ET	&
OU	|
Sauf	#

L'opérateur est optionnel et n'est pas nécessaire pour le premier appel à CHERCHER pour une recherche complexe. Il est également inutile si votre recherche s'écrit sur une seule ligne.

• Le champ est le champ sur lequel va porter la recherche. Il peut provenir d'une autre table si celle-ci est la table 1 d'une table liée à table par un lien automatique (la table à laquelle CHERCHER est appliquée doit être la table N).

• Le comparateur est l'élément qui va permettre de confronter champ et critèreRecherche. Voici la liste des comparateurs possibles :

Comparateur	Symbole à utiliser avec CHERCHER
Egal à	=
Différent de	#
Inférieur à	<
Supérieur à	>
Inférieur ou égal à	<=
Supérieur ou égal à	>=

La valeur représente ce qui va être confronté au contenu de champ. La valeur peut être toute expression du même type que champ. Le type de la valeur n'est évalué qu'une seule fois, au démarrage de la recherche, et ne l'est donc pas pour chaque enregistrement. Si la recherche porte sur le contenu d'une chaîne de caractères, utilisez dans 'valeur' le symbole "@" pour isoler le contenu à rechercher, par exemple "@Dupon@". Il est à noter, dans ce cas, que vous ne bénéficierez pas d'une recherche indexée.

Voici les règles à observer pour la construction de séquences de recherche :
• La première ligne ne doit pas contenir d'opérateur.
• Les suivantes doivent débuter par un opérateur.
• Toutes les lignes, à l'exception de la dernière, doivent s'achever par le symbole *.
• Pour lancer la recherche, ne passez pas le paramètre * à la fin de la dernière ligne. Autre solution : vous pouvez exécuter la commande CHERCHER sans autre paramètre que la table (l'Editeur de recherches ne s'affiche pas ; au lieu de cela, les lignes de recherche complexe définies auparavant sont exécutées).

Note : Chaque table maintient sa propre construction de recherche courante. Cela signifie que vous pouvez créer de multiples recherches simultanément, une pour chaque table. Dans ce cas, vous devez passer le paramètre table ou spécifier une table par défaut.

Quelle que soit la manière dont la recherche a été définie :

• Si l'exécution d'une commande CHERCHER nécessite un certain temps, 4e Dimension affiche automatiquement un message contenant un thermomètre de progression. Ces thermomètres peuvent être cachés à l'aide des commandes SUPPRIMER MESSAGES et LAISSER MESSAGES. Si le thermomètre de progression est affiché, l'utilisateur peut cliquer sur le bouton **Stop** pour interrompre l'opération. Si la recherche s'est correctement déroulée, la variable système OK prend la valeur 1. Sinon, si la recherche est interrompue, OK prend la valeur 0 (zéro).

• Si des champs indexés sont spécifiés, la recherche est optimisée à chaque fois que c'est possible (la recherche commence par les champs indexés), réduisant au maximum la durée de l'opération.

Exemples

(5) Nous recherchons tous les enregistrements dont le nom est égal à "Dupont" :

```
⇒ CHERCHER([Personnes]; [Personnes]Nom = "Dupont")
```

Note : Si le champ Nom est indexé, nous bénéficions donc d'une recherche accélérée tirant parti de l'index.

Rappel : Cette recherche trouvera les enregistrements tels que "Dupont", "dupont", "DUPONT", etc. Si vous voulez que la recherche tienne compte des majuscules/minuscules, définissez des critères supplémentaires utilisant les codes ASCII.

(6) Nous recherchons les personnes se nommant "Dupont" et se prénommant "Jean". Le champ Nom est indexé. En revanche, le champ Prénom ne l'est pas :

```
      ` Chercher toute personne qui s'appelle Dupont
⇒     CHERCHER ([Personnes]; [Personnes]Nom = "Dupont"; *)
⇒     CHERCHER ([Personnes]; & [Personnes]Prénom = "Jean")  ` et dont le prénom est Jean
```

Cet exemple effectue dans un premier temps une recherche rapide sur le champ indexé Nom, ce qui réduit la sélection d'enregistrements à ceux des personnes s'appelant Dupont. La recherche s'effectue ensuite séquentiellement sur le champ Prénom, mais nous serons peu pénalisés car elle s'exécute parmi une présélection d'enregistrements.

(7) L'exemple suivant recherche les personnes se nommant Dupont ou Blanc. Le champ Nom est indexé :

```
      ` Chercher toute personne qui s'appelle Dupont...
⇒     CHERCHER ([Personnes]; [Personnes]Nom = "Dupont"; *)
⇒     CHERCHER ([Personnes]; | [Personnes]Nom = "Blanc")  ` ou Blanc
```

La commande utilise l'index du champ Nom pour les deux recherches. Les deux recherches sont effectuées, et leurs résultats sont placés dans des ensembles internes qui sont finalement combinés par l'intermédiaire d'une opération Union.

(8) L'exemple suivant recherche des personnes qui ne travaillent pas pour une société. La recherche est effectuée en testant si le nom de la société est une chaîne vide.

```
⇒     CHERCHER ([Personnes]; [Personnes]Société = "")  ` Chercher les personnes sans société
```

(9) L'exemple suivant recherche chaque personne se nommant "Dupont" et travaillant dans une société basée à Paris. La deuxième recherche utilise un champ venant d'une autre table. Cette recherche peut être effectuée parce que la table [Personnes] est liée à la table [Société] par un lien de N vers 1 :

```
      ` Chercher toute personne qui s'appelle Dupont...
⇒     CHERCHER ([Personnes]; [Personnes]Nom = "Dupont"; *)
         ` ...qui travaille pour une société à Paris
⇒     CHERCHER ([Personnes]; & [Société]Ville = "Paris")
```

(10) L'exemple suivant recherche l'enregistrement de chaque personne dont l'initiale du nom est située entre les lettre A (incluse) et M (incluse) :

```
      ` Trouver toute personne entre A et M
⇒     CHERCHER ([Personnes]; [Personnes]Nom < "n")
```

(11) L'exemple suivant recherche les enregistrements des personnes habitant soit Paris soit Lyon :

```
      ` Trouver ceux qui habitent Paris...
⇒     CHERCHER ([Personnes]; [Personnes]CodePostal = "75@"; *)
⇒     CHERCHER ([Personnes]; | [Personnes]CodePostal = "69@")  ` ou Lyon
```

(12) Cette recherche porte sur le contenu d'un sous-champ indexé. Cette recherche ne nous retournera pas une sélection de sous-enregistrements mais une sélection d'enregistrements parents (de la table [Personnes]) dont les sous-enregistrements répondent aux critères de recherche.

```
      ` Trouver les personnes qui ont un enfant qui s'appelle Marjorie
⇒     CHERCHER ([Personnes]; [Personnes]Enfant'Nom = "Marjorie")
```

(13) Nous recherchons les enregistrements correspondant à la réponse fournie dans une boîte de dialogue :

```
      `Demander un numéro de facture à l'utilisateur
      vTrouvé := Demander ("Saisissez un numéro de facture :")
      Si (OK = 1)  ` Si l'utilisateur clique sur OK...
         `Trouver le numéro qui correspond à vTrouvé
⇒        CHERCHER ([Factures]; [Factures]Num = vTrouvé)
      Fin de si
```

(14) Cet exemple recherche tous les enregistrements des factures saisies en 1996. Nous recherchons les dates entre le 31/12/95 et le 1/1/97 :

```
      ` Trouver des factures datant d'après le 31/12/95...
⇒     CHERCHER ([Factures]; [Factures]DateFacture > !31/12/95!; *)
⇒     CHERCHER ([Factures]; & [Factures]DateFacture < !1/1/97!) ` et d'avant le 1/1/97
```

(15) L'exemple suivant trouve les employés qui ont un salaire entre 100 000 F et 250 000 F. La recherche inclut les employés qui gagnent 100 000 F et exclut ceux qui gagnent 250 000 F :

```
      ` Trouver les employés qui ont un salaire entre...
⇒     CHERCHER ([Employés]; [Employés]Salaire >= 100000; *)
⇒     CHERCHER ([Employés]; & [Employés]Salaire < 250000)  ` 100 000 F et 250 000 F
```

(16) L'exemple suivant cherche les employés du service Marketing qui ont un salaire supérieur à 150 000 F. Le champ Salaire est utilisé dans un premier temps car il est indexé. Notez que la seconde recherche utilise un champ venant d'une autre table. Le champ [Service]Nom est lié à la table [Employés] par un lien automatique de N vers 1. Le champ [Service]Nom est indexé, mais la recherche n'est pas indexée car le lien doit être activé séquentiellement pour chaque enregistrement dans la table [Employés] :

```
      ` Trouver les employés qui ont un salaire supérieur à 150 000 F
⇒     CHERCHER ([Employés]; [Employés]Salaire > 20000; *)
      ` et qui travaillent dans le service marketing
⇒     CHERCHER ([Employés]; & [Service]Nom = "marketing")
```

(17) La recherche suivante recherche les informations égales à la valeur de la variable mavar.

```
      ` Trouver toutes les lois qui sont égales à la valeur de mavar
⇒     CHERCHER ([Lois]; [Lois]Texte = mavar)
```

La recherche peut avoir des résultats différents selon la valeur de mavar. Elle sera également exécutée différement. Par exemple :
• Si mavar est égale à "Copyright@", la sélection contient toutes les lois qui commencent par Copyright.
• Si mavar est égale à "@Copyright@", la sélection contient toutes les lois qui contiennent au moins une occurrence de Copyright.

Référence

CHERCHER DANS SELECTION.

CHERCHER DANS SELECTION ({table}{; critère}{; *})

Paramètre	Type		Description
table	Table	→	Table dans laquelle effectuer la recherche ou ou Table par défaut si ce paramètre est omis
critère		→	Lignes de recherche
*		→	Attente d'exécution de la recherche

Description

CHERCHER DANS SELECTION recherche des enregistrements dans table. CHERCHER DANS SELECTION modifie la sélection courante de table pour le process courant. Le premier enregistrement de la nouvelle sélection devient l'enregistrement courant.

CHERCHER DANS SELECTION a un fonctionnement et des effets proches de ceux de CHERCHER. La différence entre ces deux commandes est la portée de la recherche :
• CHERCHER recherche des enregistrements dans la table.
• CHERCHER DANS SELECTION recherche des enregistrements parmi la sélection courante de la table.

Pour plus d'informations, reportez-vous à la description de la commande CHERCHER.

Note : La commande FIXER PARAMETRE BASE permet de définir si CHERCHER DANS SELECTION doit utiliser ou non l'index, en fonction du nombre d'enregistrements présents dans la sélection.

Exemple

Cet exemple illustre la différence entre CHERCHER et CHERCHER DANS SELECTION. Voici deux recherches :

```
      ` Trouver TOUTES les sociétés basées à Paris
CHERCHER ([Sociétés]; [Sociétés]Ville="Paris")
   ` Trouver toutes les sociétés s'occupant d'affaires boursières (quelle que soit leur ville)
CHERCHER ([Sociétés]; [Sociétés]Activité="Affaires boursières")
```

Notez que le second CHERCHER "ignore" complètement les résultats du premier. Comparez avec :

```
                ` Trouver TOUTES les sociétés basées à Paris
        CHERCHER ([Sociétés]; [Sociétés]Ville="Paris")
                ` Trouver TOUTES les sociétés s'occupant d'affaires boursières basées à Paris
⇒       CHERCHER DANS SELECTION ([Sociétés]; [Sociétés]Activité="Affaires boursières")
```

CHERCHER DANS SELECTION n'effectue sa recherche que parmi les enregistrements sélectionnés, dans cet exemple les sociétés basées à Paris.

Référence

CHERCHER, FIXER PARAMETRE BASE.

CHERCHER PAR FORMULE ({table}{; }{formule})

Paramètre	Type		Description
table	Table	→	Table dans laquelle effectuer la recherche ou Table par défaut si ce paramètre est omis
formule	Booléen	→	Formule de recherche

Description

CHERCHER PAR FORMULE effectue une recherche d'enregistrements dans table. CHERCHER PAR FORMULE modifie la sélection courante de table pour le process courant et fait du premier enregistrement le nouvel enregistrement courant.

CHERCHER PAR FORMULE et la commande CHERCHER PAR FORMULE DANS SELECTION fonctionnent exactement de la même manière, à la différence près que CHERCHER PAR FORMULE effectue sa recherche parmi la totalité des enregistrements de la table alors que CHERCHER PAR FORMULE DANS SELECTION se cantonne aux enregistrements de la sélection courante.

Les deux commandes appliquent formule à chaque enregistrement de la table ou de la sélection. formule est une expression booléenne qui doit retourner VRAI ou FAUX. Si formule retourne Vrai, l'enregistrement est inclus dans la nouvelle sélection.

formule peut être simple (par exemple la comparaison d'un champ à une valeur) ou complexe (réalisation d'un calcul ou même évaluation de valeurs dans une table liée). Ce peut être une fonction 4e Dimension, ou une fonction ou une expression que vous avez créée. Vous pouvez utiliser des jokers dans formule lorsque vous travaillez avec des champs de type Alpha ou Texte.

Si vous omettez le paramètre formule, 4D affiche la boîte de dialogue standard de recherche.

Lorsque la recherche est terminée, le premier enregistrement de la nouvelle sélection est chargé depuis le disque et devient l'enregistrement courant.

Ces commandes effectuent toujours une recherche séquentielle, non indexée. CHERCHER PAR FORMULE et CHERCHER PAR FORMULE DANS SELECTION sont plus lentes que CHERCHER lorsqu'elles sont utilisées sur des champs indexés. La durée de la recherche est proportionnelle au nombre d'enregistrements présents dans la table ou la sélection.

4D Server : Le serveur n'exécute pas la formule de recherche. Chaque enregistrement est envoyé au poste client et la formule de recherche est exécutée sur le client. Ce fonctionnement rend ces commandes moins efficaces avec 4D Server que la commande CHERCHER.

Exemples

(1) L'exemple suivant recherche les enregistrements de toutes les factures qui ont été saisies au mois de décembre, sans tenir compte de l'année. Le principe est d'appliquer la fonction Mois de à chaque enregistrement. Cette recherche ne pourrait pas être effectuée d'une autre manière sans créer un champ séparé pour le mois :

```
⇒ CHERCHER PAR FORMULE ([Factures]; Mois de ([Factures]Saisie) = 12)
   ` Chercher les factures saisies en décembre
```

(2) L'exemple suivant recherche les enregistrements de toutes les personnes dont le nom comporte plus de dix caractères :

```
⇒ CHERCHER PAR FORMULE ([Personnes]; Longueur ([Personnes]Nom) > 10)
   ` Chercher les personnes dont le nom fait plus de dix caractères
```

Référence

CHERCHER, CHERCHER DANS SELECTION, CHERCHER PAR FORMULE DANS SELECTION.

CHERCHER PAR FORMULE DANS SELECTION ({table}{; }{formule})

Paramètre	Type		Description
table	Table	→	Table dans laquelle effectuer la recherche parmi la sélection courante ou Table par défaut si ce paramètre est omis
formule	Booléen	→	Formule de recherche

Description

La commande CHERCHER PAR FORMULE DANS SELECTION vous permet de rechercher des enregistrements dans table. CHERCHER PAR FORMULE DANS SELECTION modifie la sélection courante de table pour le process courant et fait du premier enregistrement le nouvel enregistrement courant.

CHERCHER PAR FORMULE DANS SELECTION fonctionne de la même manière que CHERCHER PAR FORMULE. La différence entre ces deux commande se situe au niveau de la portée de la recherche :
• CHERCHER PAR FORMULE effectue sa recherche parmi la totalité des enregistrements de la table.
• CHERCHER PAR FORMULE DANS SELECTION effectue sa recherche uniquement parmi les enregistrements de la sélection courante.

Pour plus d'informations, reportez-vous à la description de la commande CHERCHER PAR FORMULE.

Référence

CHERCHER, CHERCHER DANS SELECTION, CHERCHER PAR FORMULE.

CHERCHER PAR TABLEAU (champIndexé; tableau)

Paramètre	Type		Description
champIndexé	Champ	→	Champ indexé duquel comparer les valeurs
tableau	Tableau	→	Tableau des valeurs recherchées

Description

La commande CHERCHER PAR TABLEAU recherche dans la table du champ passé en premier paramètre tous les enregistrements pour lesquels la valeur de champIndexé est égale à au moins une des valeurs des éléments du tableau tableau. Les enregistrements trouvés constituent la nouvelle sélection courante.

Cette commande permet de construire rapidement et simplement une recherche sur plusieurs valeurs.

Notes :
• Cette commande ne fonctionne qu'avec des champs indexés. Elle ne peut donc pas être utilisée avec des champs de type texte, image, sous-table et BLOB.
• Rappelons qu'un tableau de type Entier long est compatible avec un champ de type Heure.

Exemple

Cet exemple permet de récupérer les enregistrements des clients français et américains :

```
   TABLEAU ALPHA (2;TabRecherche;30)
   TabRecherche{1}:="FR"
   TabRecherche{2}:="US"
⇒  CHERCHER PAR TABLEAU ([Clients]Pays;TabRecherche)
```

FIXER DESTINATION RECHERCHE (destinationType{; destinationObjet})

Paramètre	Type		Description
destinationType	Numérique	→	0 sélection courante 1 ensemble 2 sélection temporaire 3 variable
destinationObjet	Chaine | Variable	→	Nom de l'ensemble ou de la sélection temporaire ou Variable

Description

La commande FIXER DESTINATION RECHERCHE vous permet d'indiquer à 4e Dimension où placer les résultats de toutes les recherches qui suivent l'appel de cette commande dans le process courant.

Vous spécifiez le type de la destination dans le paramètre destinationType. 4e Dimension fournit les constantes prédéfinies suivantes :

Constante	Type	Valeur
Vers sélection courante	Entier long	0
Vers ensemble	Entier long	1
Vers sélection temporaire	Entier long	2
Vers variable	Entier long	3

Vous spécifiez le nom de la destination de la recherche dans le paramètre optionnel destinationObjet en fonction du tableau suivant :

Paramètre destinationType	Paramètre destinationObjet
0 (sélection courante)	Vous ne passez pas de paramètre.
1 (ensemble)	Vous passez le nom de l'ensemble (existant ou à créer)
2 (sélection temporaire)	Vous passez le nom de la sélection temporaire (existante ou à créer)
3 (variable)	Vous passez une variable numérique (existante ou à créer)

Avec

FIXER DESTINATION RECHERCHE(Vers sélection courante)

Les enregistrements trouvés par la recherche seront placés dans la sélection courante de la table dans laquelle la recherche est effectuée.

Avec

FIXER DESTINATION RECHERCHE(Vers ensemble;"monEnsem")

Les enregistrements trouvés par la recherche seront placés dans l'ensemble monEnsem. La sélection courante et l'enregistrement courant de la table dans laquelle vous recherchez restent inchangés.

Avec

FIXER DESTINATION RECHERCHE(Vers sélection temporaire;"mTemp")

Les enregistrements trouvés par la recherche seront placés dans la sélection temporaire maTemp. La sélection courante et l'enregistrement courant pour la table sur laquelle vous effectuez la recherche restent inchangés.

Avec

FIXER DESTINATION RECHERCHE(Vers variable;$vlRésultatRech)

Le **nombre** d'enregistrements trouvés par la recherche sera placé dans la variable $vlRésultatRech. La sélection courante et l'enregistrement courant de la table dans laquelle vous effectuez la recherche restent inchangés.

Attention : FIXER DESTINATION RECHERCHE affecte toutes les recherches suivantes dans le process courant. N'oubliez pas d'associer toujours un appel à FIXER DESTINATION RECHERCHE (lorsque destinationType#0) à un appel à FIXER DESTINATION RECHERCHE(0) ultérieur pour rétablir le mode standard de recherche.

FIXER DESTINATION RECHERCHE modifie uniquement le comportement des commandes de recherche, c'est-à-dire :

- CHERCHER
- CHERCHER DANS SELECTION
- CHERCHER PAR EXEMPLE
- CHERCHER PAR FORMULE
- CHERCHER PAR FORMULE DANS SELECTION
- CHERCHER PAR TABLEAU

En revanche, FIXER DESTINATION RECHERCHE n'affecte pas les autres commandes qui modifient la sélection courante telles que TOUT SELECTIONNER, LIEN RETOUR, etc.

Exemples

(1) Vous créez un formulaire qui affiche les enregistrements venant de la table [Annuaire]. Vous créez un objet du type onglet qui s'appelle asRolodex (avec un onglet pour chaque lettre dans l'alphabet) et un sous-formulaire qui affiche les enregistrements de la table [Annuaire]. En choisissant un onglet, vous affichez les enregistrements qui correspondent à cette lettre. Puisque, dans cet exemple, la table [Annuaire] contient des données statiques, vous ne voulez pas effectuer une recherche chaque fois que vous cliquez sur un onglet et donc vous dépensez moins de temps précieux à exécuter ces recherches. Pour faire ceci, vous pouvez placer vos recherches dans les sélections temporaires pour les réutiliser quand il le faut. Vous écrivez la méthode objet de l'onglet asRolodex comme indiquée ci-dessous :

```
    ` méthode objet de l'onglet asRolodex
Au cas ou

: (Evenement formulaire=Sur chargement)
      ` Avant que le formulaire s'affiche à l'écran,
      ` initialiser l'onglet et le tableau de booléens qui nous indiquent
      ` si une recherche pour la lettre sur laquelle vous avez cliqué
      ` a été exécutée ou pas
   TABLEAU ALPHA(1;asRolodex;26)
   TABLEAU BOOLEEN(abRechFini;26)
   Boucle ($vlElém;1;26)
      asRolodex{$vlElém}:=Caractere(64+$vlElém)
      abRechFini{$vlElém}:=Faux
   Fin de boucle

: (Evenement formulaire=Sur clic)
      ` Lorsque l'utilisateur clique sur un onglet, vérifier si une recherche pour cette
      ` lettre a été exécutée ou non
   Si (Non(abRechFini{asRolodex}))
         ` Sinon, fixer la destination de la recherche vers une sélection temporaire
⇒     FIXER DESTINATION RECHERCHE(Vers sélection temporaire;"Rolodex"+
                                                    asRolodex{asRolodex})
      ` Effectuer la recherche
      CHERCHER([Phone Book];[Phone Book]Last name=asRolodex{asRolodex}+"@")
      ` Restituer le mode standard de recherche
⇒     FIXER DESTINATION RECHERCHE(Vers sélection courante)
      ` La prochaine fois que cette lettre sera choisie, la recherche ne sera
      ` pas exécutée
      abRechFini{asRolodex}:=Vrai
   Fin de si
      ` Utiliser la sélection temporaire pour l'affichage des enregistrements qui
      ` correspondent à cette lettre
   UTILISER SELECTION("Rolodex"+asRolodex{asRolodex})
```

```
: (Evenement formulaire=Sur libération)
      ` Après que le formulaire disparaît de l 'écran
      ` Effacer les sélections temporaires de la mémoire
   Boucle (vlElem;1;26)
      Si(abRechFini{vlElém})
         EFFACER SELECTION("Rolodex"+asRolodex{vlElém})
      Fin de si
   Fin de boucle
      ` Effacer les deux tableaux dont nous n'avons pas besoin
   EFFACER VARIABLE(asRolodex)
   EFFACER VARIABLE(abRechFini)
Fin de cas
```

(2) La méthode ValeursUniques suivante vérifie si les valeurs sont uniques pour des champs dans une table de votre choix. L'enregistrement courant peut déjà exister ou vient d'être créé.

```
   ` Méthode projet ValeursUniques
   ` ValeursUniques ( Pointeur ; Pointeur { ; Pointeur... } ) -> Booléen
   ` ValeursUniques ( ->Table ; ->Champ { ; ->Champ2...  } ) -> Oui ou non

C_BOOLEEN($0;$2)
C_POINTEUR(${1})
C_ENTIER LONG($vlChamp;$vlNmbChamps;$vlTrouvé;$vlEnregCour)
$vlNmbChamps:=Nombre de parametres-1
$vlEnregCour:=Numero enregistrement($1->)
Si ($vlNmbChamps>0)
   Si ($vlEnregCour#-1)
      Si ($vlEnregCour<0)
            ` Il s'agit d'un nouvel enregistrement qui n'a pas été sauvegardé (numéro
            ` d'enregistrement est égal à -3), donc nous pouvons arrêter
            ` la recherche dès que nous avons trouvé un enregistrement
⇒        FIXER LIMITE RECHERCHE(1)
      Sinon
            ` Il s'agit d'un enregistrement existant, donc nous pouvons arrêter
            ` la recherche dès que nous avons trouvé au moins deux enregistrements
⇒        FIXER LIMITE RECHERCHE(2)
      Fin de si
         ` La recherche retournera le résultat dans la variable $vlTrouvé
         ` sans changer l'enregistrement courant ni la sélection courante
⇒     FIXER DESTINATION RECHERCHE(Vers variable;$vlTrouvé)
         ` Construire la recherche selon le nombre de champs spécifiés
      Au cas ou
         : ($vlNmbChamps=1)
            CHERCHER($1->;$2->=$2->)
         : ($vlNmbChamps=2)
            CHERCHER($1->;$2->=$2->;*)
            CHERCHER($1->; & ;$3->=$3->)
```

```
            Sinon
               CHERCHER($1->;$2->=$2->;*)
               Boucle ($vlChamp;2;$vlNmbChamps-1)
                  CHERCHER($1->; & ;${1+$vlChamp}->=${1+$vlChamp}->;*)
               Fin de boucle
               CHERCHER($1->; & ;${1+$vlNmbChamps}->=${1+$vlNmbChamps}->)
            Fin de cas
⇒           FIXER DESTINATION RECHERCHE(0) ` Rétablir le mode standard de recherche
⇒           FIXER LIMITE RECHERCHE(0) ` Enlever la limite sur la recherche
            ` Traiter le résultat de la recherche
            Au cas ou
               : ($vlTrouvé=0)
                  $0:=Vrai ` Pas de valeurs dupliquées
               : ($vlTrouvé=1)
                  Si ($vlEnregCour<0)
                        ` Un enregistrement existe, avec les mêmes valeurs que le nouveau
                     $0:=Faux
                  Sinon
                        ` Pas de valeurs dupliquées, c'est le même enregistrement
                     $0:=Vrai
                  Fin de si
               : ($vlTrouvé=2)
                  $0:=Faux ` Quoi que ce soit, les valeurs sont dupliquées
            Fin de cas
         Sinon
            ` Cela n'a pas de sens, signalez-le pendant le développement de la base
            Si (◊Débogage)
               ` ATTENTION ! Cette méthode a été appelée sans enregistrement courant
               TRACE
            Fin de si
            $0:=Faux ` Ne peut pas garantir le résultat
         Fin de si
      Sinon
         ` Cela n'a pas de sens, signalez-le pendant le développement de la base
         Si (◊Débogage)
            ` ATTENTION ! Cette méthode a été appelée sans conditions de recherche
            TRACE
         Fin de si
         $0:=Faux ` Ne peut pas garantir le résultat
      Fin de si
```

Lorsque cette méthode est implémentée dans votre application, vous pouvez écrire le code suivant :

```
` ...
Si (ValeursUniques (->[Contacts];->[Contacts]Société;->[Contacts]Nom;
                                                    ->[Contacts]Prénom))
   ` Traitement de l'enregistrement qui a les valeurs uniques
Sinon
   ALERTE("Il existe déjà un contact avec ce nom pour cette société.")
Fin de si
` ...
```

Référence

CHERCHER, CHERCHER DANS SELECTION, CHERCHER PAR EXEMPLE, CHERCHER PAR FORMULE, CHERCHER PAR FORMULE DANS SELECTION, FIXER LIMITE RECHERCHE.

FIXER LIMITE RECHERCHE (limite)

Paramètre	Type		Description
limite	Numérique	→	Nombre limite d'enregistrements ou 0 pour nombre illimité

Description

La commande FIXER LIMITE RECHERCHE vous permet d'indiquer à 4e Dimension d'arrêter toutes les recherches suivant l'appel de cette commande dans le process courant dès que le nombre d'enregistrements défini dans limite a été atteint.

Si, par exemple, limite est égal à 1, les recherches s'arrêteront dès qu'un enregistrement sera trouvé selon les conditions de la recherche.

Pour que les recherches soient de nouveau sans limite, appelez FIXER LIMITE RECHERCHE en fixant le paramètre limite à 0.

Attention : FIXER LIMITE RECHERCHE affecte toutes les recherches dans le process courant. N'oubliez pas d'associer toujours un appel à FIXER LIMITE RECHERCHE(limite) (lorsque limite>0) à un appel à FIXER LIMITE RECHERCHE(0) ultérieur pour rétablir les recherches sans limite.

FIXER LIMITE RECHERCHE modifie uniquement le comportement des commandes de recherche, c'est-à-dire :

- CHERCHER
- CHERCHER DANS SELECTION
- CHERCHER PAR EXEMPLE
- CHERCHER PAR FORMULE
- CHERCHER PAR FORMULE DANS SELECTION
- CHERCHER PAR TABLEAU

En revanche, FIXER LIMITE RECHERCHE n'affecte pas les autres commandes qui modifient la sélection courante d'une table telles que TOUT SELECTIONNER, LIEN RETOUR, etc.

Exemples

(1) Pour effectuer une recherche qui correspond à la formule "...trouver dix clients avec lesquels les ventes sont supérieures à 1MF...", écrivez le code suivant :

```
⇒ FIXER LIMITE RECHERCHE(10)
  CHERCHER([Clients];[Clients]Ventes>1000000)
⇒ FIXER LIMITE RECHERCHE(0)
```

(2) Référez-vous au deuxième exemple de la commande FIXER DESTINATION RECHERCHE.

Référence

CHERCHER, CHERCHER DANS SELECTION, CHERCHER PAR EXEMPLE, CHERCHER PAR FORMULE, CHERCHER PAR FORMULE DANS SELECTION, FIXER DESTINATION RECHERCHE.

Trouver clef index (champIndexé; valeur) → Entier long

Paramètre	Type		Description
champIndexé	Champ	→	Champ indexé sur lequel effectuer la recherche
valeur		→	Valeur à rechercher
		←	Valeur trouvée
Résultat	Entier long	←	Numéro de l'enregistrement trouvé ou -1 si pas d'enregistrement trouvé

Description

La commande Trouver clef index retourne le numéro du premier enregistrement dont le champ champIndexé est égal à la valeur valeur.
Si aucun enregistrement ne correspond au critère, Trouver clef index retourne -1.

Après l'appel, le paramètre valeur contient la valeur effectivement trouvée. Ce fonctionnement permet d'effectuer des recherches utilisant le caractère "@" sur des champs de type alpha, et pour lesquelles il est nécessaire de récupérer la valeur trouvée.

La commande ne modifie ni la sélection courante, ni l'enregistrement courant.

Cette fonction, très rapide car exploitant uniquement l'index, est particulièrement utile pour prévenir la création de doublons au moment de la saisie de données.

Exemple

Dans une base de données de CD audio, vous souhaitez vérifier, au moment de la saisie d'un nouveau nom de chanteur, si celui-ci n'existe pas déjà dans la base. Comme il peut exister des homonymes, vous ne souhaitez pas toutefois que le champ [Chanteur]Nom soit unique. Pour cela, dans le formulaire d'entrée, vous écrivez dans la méthode objet du champ [Chanteur]Nom :

```
   Si (Evenement formulaire=Sur données modifiées)
⇒     $EnrgNum:=Trouver clef index([Chanteur]Nom;[Chanteur]Nom)
      Si ($EnrgNum # -1)  ` Si ce nom a déjà été saisi
         CONFIRMER("Un chanteur de ce nom existe déjà. Voulez-vous visualiser
                                                              sa fiche ?";"Oui";"Non")
         Si (OK=1)
            ALLER A ENREGISTREMENT([Chanteur];$EnrgNum)
         Fin de si
      Fin de si
   Fin de si
```

TRIER ({table}{; champ}{; > ou <}{; champ2; > ou <2; ...; champN; > ou <N}{; *})

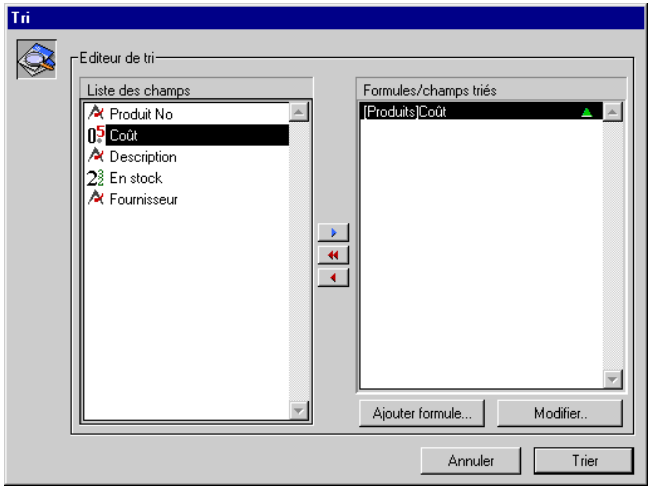
Paramètre	Type		Description
table	Table	→	Table de laquelle réordonner la sélection courante ou Table par défaut si ce paramètre est omis
champ	Champ	→	Champ sur lequel effectuer le tri pour chaque niveau
> ou <		→	Sens du tri pour chaque niveau : > demander un tri croissant ou < demander un tri décroissant
*		→	Attente d'exécution du tri

Description

TRIER trie (réordonne) les enregistrements de la sélection courante de table pour le process courant. Une fois le tri effectué, le premier enregistrement de la nouvelle sélection courante devient le nouvel enregistrement courant.

Si vous omettez le paramètre table, la commande s'applique à la table par défaut. Si aucune table par défaut n'a été définie, une erreur est retournée.

Si vous ne passez ni le paramètre champ, ni les paramètres >, < ou *, TRIER affiche la boîte de dialogue de l'Editeur de tri de 4D pour table. Cet éditeur du mode Utilisation est présenté ci-dessous :



Pour plus d'informations sur l'utilisation de cet éditeur, reportez-vous au manuel *Mode Utilisation* de 4D.
L'utilisateur construit le tri puis clique sur le bouton **Trier**. Si le tri est correctement effectué, la variable système OK prend la valeur 1. Si l'utilisateur clique sur **Annuler**, aucun tri n'est effectué et la variable OK prend la valeur 0 (zéro).

Exemples

(1) L'exemple suivant affiche la boîte de dialogue de Tri pour la table [Produits] :

⇒ **TRIER** ([Produits])

(2) L'exemple suivant affiche la boîte de dialogue de Tri pour la table par défaut (si elle a été définie) :

⇒ **TRIER**

Si vous spécifiez les paramètres champ et > ou <, la boîte de dialogue standard de Tri ne s'affiche pas et le tri est entièrement défini par programmation. Vous pouvez trier la sélection courante sur un plusieurs niveaux. Pour chaque niveau de tri, vous passez un champ dans le paramètre champ et un ordre de tri dans > ou <. Si vous passez le paramètre “supérieur à ” (>), l'ordre est croissant. Si vous passez le paramètre “inférieur à ” (<), l'ordre est décroissant.

Exemples

(3) L'exemple suivant trie la sélection courante de [Produits] par nom dans un ordre croissant :

⇒ **TRIER** ([Produits]; [Produits]Nom;>)

(4) L'exemple suivant trie la sélection courante de [Produits] par nom dans un ordre décroissant :

⇒ **TRIER** ([Produits]; [Produits]Nom;<)

(5) L'exemple suivant trie la sélection courante de [Produits] par type et par prix dans un ordre croissant à chaque niveau :

⇒ **TRIER** ([Produits]; [Produits]Type;>; [Produits]Prix;>)

(6) L'exemple suivant trie la sélection courante de [Produits] par type et par prix dans un ordre décroissant à chaque niveau :

⇒ **TRIER** ([Produits]; [Produits]Type;<; [Produits]Prix;<)

(7) L'exemple suivant trie la sélection courante de [Produits] par type dans un ordre croissant et par prix dans un ordre décroissant :

⇒ **TRIER** ([Produits]; [Produits]Type;>; [Produits]Prix;<)

(8) L'exemple suivant trie la sélection courante de [Produits] par type dans un ordre décroissant et par prix dans un ordre croissant :

⇒ **TRIER** ([Produits]; [Produits]Type;<; [Produits]Prix;>)

Si vous omettez le paramètre d'ordre > ou <, le tri est croissant par défaut.

Exemple
(9) L'exemple suivant trie la sélection courante de [Produits] par nom dans un ordre croissant :

⇒ **TRIER** ([Produits]; [Produits]Nom)

Si un seul champ est spécifié (tri sur un niveau) et s'il est indexé, le tri tire parti de l'index. Si le champ n'est pas indexé ou si plus d'un champ est utilisé, le tri est effectué de manière séquentielle. Le champ peut appartenir à la table de la sélection que vous triez ou à une table 1 liée à table par un lien automatique (la table réordonnée doit être la table N). Dans ce cas, le tri est toujours séquentiel.

Exemples
(10) L'exemple suivant effectue un tri indexé si le champ [Produits]Nom est indexé :

⇒ **TRIER** ([Produits]; [Produits]Nom;>)

(11) L'exemple suivant effectue un tri séquentiel, que les champs soient ou non indexés :

⇒ **TRIER** ([Produits]; [Produits]Type;>;[Produits]Prix;>)

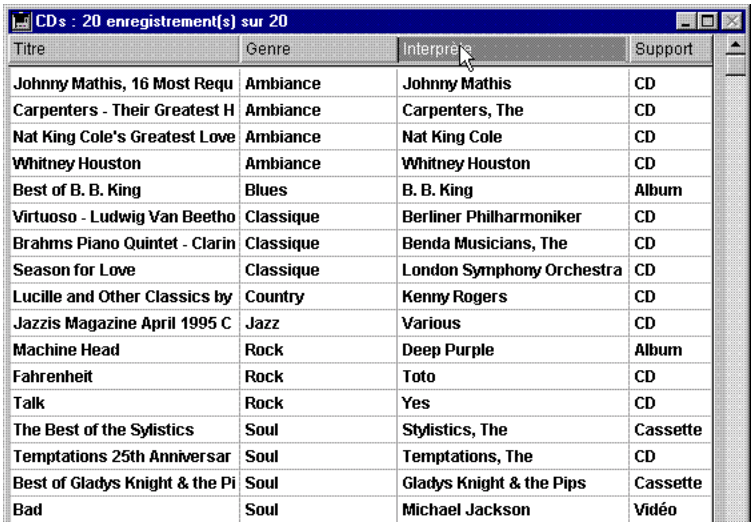
(12) L'exemple suivant effectue un tri séquentiel à l'aide d'un champ lié :

⇒ **TRIER** ([Factures];[Société]Nom;>) ` les factures sont triées par ordre alphabétique sur le champ Nom de la société

Pour indiquer que le tri ne doit pas être immédiatement effectué, passez en dernier paramètre le symbole *. 4e Dimension attendra de rencontrer une nouvelle ligne de tri ne se terminant pas par * pour exécuter le tri. Cette possibilité est utile pour gérer les tris multicritères dans le cadre d'interfaces personnalisées.
Attention : lorsque vous utilisez cette syntaxe, vous ne pouvez passer qu'un seul niveau de tri (un seul champ) par ligne d'instruction.

Exemple

(13) Dans un formulaire sortie affiché en mode Menus créés, vous souhaitez permettre aux utilisateurs de trier une colonne par ordre croissant en cliquant sur son en-tête. Si l'utilisateur maintient la touche **Maj** enfoncée et clique ensuite sur plusieurs autres colonnes, le tri est multicritères, c'est-à-dire que les colonnes sont triées sur autant de niveaux qu'il y a de clics :



Titre	Genre	Interprète	Support
Johnny Mathis, 16 Most Requ	Ambiance	Johnny Mathis	CD
Carpenters - Their Greatest H	Ambiance	Carpenters, The	CD
Nat King Cole's Greatest Love	Ambiance	Nat King Cole	CD
Whitney Houston	Ambiance	Whitney Houston	CD
Best of B. B. King	Blues	B. B. King	Album
Virtuoso - Ludwig Van Beetho	Classique	Berliner Philharmoniker	CD
Brahms Piano Quintet - Clarin	Classique	Benda Musicians, The	CD
Season for Love	Classique	London Symphony Orchestra	CD
Lucille and Other Classics by	Country	Kenny Rogers	CD
Jazzis Magazine April 1995 C	Jazz	Various	CD
Machine Head	Rock	Deep Purple	Album
Fahrenheit	Rock	Toto	CD
Talk	Rock	Yes	CD
The Best of the Sylistics	Soul	Stylistics, The	Cassette
Temptations 25th Anniversar	Soul	Temptations, The	CD
Best of Gladys Knight & the Pi	Soul	Gladys Knight & the Pips	Cassette
Bad	Soul	Michael Jackson	Vidéo

Chaque en-tête de colonne contient un bouton inversé dont la méthode est du type suivant :

```
MULTITRIS (->[CDs]Titre)  `Bouton de l'en-tête de la colonne Titre
```

Chaque bouton appelle la méthode projet *MULTITRIS* en passant un pointeur sur le champ de la colonne. Voici le contenu de la méthode projet *MULTITRIS* :

```
` Méthode projet MULTITRIS
` MULTITRIS (Pointeur)
` MULTITRIS (->[Table]Champ)

C_POINTEUR($1)
C_ENTIER LONG($nbCrit)

`Construction des critères
Si (Non(Majuscule enfoncee))  `Si le tri est simple
   TABLEAU POINTEUR(tPtrTriChp;1)  `Créons un tableau à 1 élément
   tPtrTriChp{1}:=$1  `Champ sur lequel l'utilisateur a cliqué
```

```
      Sinon  `Si la touche Maj était enfoncée (tri multicritère)
                `Vérifions que le critère n'est pas déjà présent
         $nbCrit:=Chercher dans tableau(tPtrTriChp;$1)
         Si ($nbCrit<0)   `Critère inexistant
               `Remplissons le tableau
            INSERER LIGNES(tPtrTriChp;Taille tableau(tPtrTriChp)+1;1)
            tPtrTriChp{Taille tableau(tPtrTriChp)}:=$1
         Fin de si
      Fin de si
         `Exécution du tri
      $nbCrit:=Taille tableau(tPtrTriChp)
      Si ($nbCrit>0)  `S'il y a au moins un élément dans le tableau de pointeurs
         Boucle ($i;1;$nbCrit)   `Pour chaque critère défini
⇒           TRIER([CDs];(tPtrTriChp{$i})->;>;*)  `On construit le tri
         Fin de boucle
⇒        TRIER([CDs])  `Pas de * : on effectue le tri
      Fin de si
```

Quelle que soit la manière dont le tri est défini, si l'opération risque de prendre un certain temps, 4e Dimension affiche automatiquement un message contenant un thermomètre de progression. Vous pouvez décider d'afficher ou de ne pas afficher ce message pour le process à l'aide des commandes SUPPRIMER MESSAGES et LAISSER MESSAGES. Si le thermomètre de progression est affiché, l'utilisateur peut cliquer sur le bouton **Stop** pour interrompre l'opération. Si le tri s'est correctement déroulé, la variable système OK prend la valeur 1. Sinon, si le tri est interrompu, OK prend la valeur 0 (zéro).

Référence

TRIER PAR FORMULE.

TRIER PAR FORMULE (table{; expression}{; > ou <}{; expression2; > ou <2; ...; expressionN; > ou <N})

Paramètre	Type		Description
table	Table	→	Table de laquelle trier la sélection d'enregistrements
expression		→	Formule de tri des enregistrements (peut être de type Alphanumérique, Réel, Entier, Entier long, Date, Heure ou Booléen)
> ou <		→	Ordre de tri pour chaque niveau : > ordre croissant ou < ordre décroissant

Description

TRIER PAR FORMULE trie (réordonne) les enregistrements de la sélection courante de table pour le process courant. Une fois le tri effectué, le premier enregistrement de la nouvelle sélection courante devient le nouvel enregistrement courant.

Notez que vous devez spécifier table. Vous ne pouvez pas utiliser une table par défaut.

Vous pouvez trier la sélection sur un ou plusieurs niveaux. Pour chaque niveau, vous passez une expression dans expression et un ordre de tri dans > ou <. Si vous passez le symbole “supérieur à” (>), l'ordre est croissant. Si vous passez le symbole “inférieur à” (<), l'ordre est décroissant. Si vous ne passez pas ce paramètre, l'ordre est par défaut croissant.

Le paramètre expression peut être de type Alpha, Réel (Numérique), Entier, Entier long, Date, Heure ou Booléen.

Quelle que soit la manière dont le tri est défini, si l'opération risque de prendre un certain temps, 4e Dimension affiche automatiquement un message contenant un thermomètre de progression. Vous pouvez décider d'afficher ou de ne pas afficher ce message pour le process à l'aide des commandes SUPPRIMER MESSAGES et LAISSER MESSAGES. Si le thermomètre de progression est affiché, l'utilisateur peut cliquer sur le bouton **Stop** pour interrompre l'opération. Si le tri s'est correctement déroulé, la variable système OK prend la valeur 1. Sinon, si le tri est interrompu, OK prend la valeur 0 (zéro).

4D Server : Comme expression ne peut pas être interprétée par 4D Server, chaque enregistrement est envoyé au poste client et la formule de tri est évaluée sur ce poste. Pour cette raison, le tri par formule n'est guère efficace, il est conseillé d'utiliser de préférence la commande TRIER lorsque c'est possible.

A la différence de TRIER, TRIER PAR FORMULE effectue toujours un tri séquentiel.

Exemple

L'exemple suivant trie les enregistrements de la table [Personnes] dans l'ordre décroissant par rapport à la longueur du nom de famille de chaque personne. L'enregistrement de la personne qui a le nom le plus long sera le premier enregistrement de la sélection courante :

⇒ **TRIER PAR FORMULE** ([Personnes]; **Longueur** ([Personnes]Nom); <)

Référence

TRIER.

45

Ressources

Qu'est-ce qu'une ressource ?

Une **ressource** regroupe des données de tout type, structurées dans un format défini, et stockées dans un fichier Windows **.RSR** ou dans la **resource fork** ("partie de ressources") d'un fichier MacOS. Généralement, les ressources contiennent des chaînes de caractères, des images, des icônes, etc. En fait, vous pouvez créer et utiliser vos propres types de ressources et y stocker toutes les données que vous voulez.

Data fork et Resource fork

Sur Macintosh, chaque fichier peut contenir une **data fork** ("partie de données") et une **resource fork** ("partie de ressources"). La data fork d'un fichier Macintosh est l'équivalent d'un fichier Windows ou UNIX. La resource fork d'un fichier Macintosh contient les ressources spécifiques MacOS du fichier et n'a pas d'équivalent direct sous Windows ou UNIX.

Sous Windows, les ressources spécifiques Windows sont intégrées aux autres données du fichier. Par exemple, une application Windows —un fichier .EXE— peut contenir à la fois des données de type ressource et du code. Afin d'assurer l'indépendance de plate-forme de vos applications 4D, 4e Dimension exploite les ressources de type MacOS sur les plates-formes Macintosh et Windows.

4D Transporter

Comme les resource forks n'existent pas sous Windows, le programme utilitaire **4D Transporter** (fourni avec la version Macintosh de 4e Dimension) vous permet de transporter une base 4D de Windows vers MacOS et inversement.

Lorsque vous transportez une base 4D de Windows vers MacOS, les fichiers .4DB et .RSR de la structure de la base sont **fusionnés** en un seul fichier Macintosh. Le fichier .4DB devient la data fork du fichier de structure Macintosh, et le fichier .RSR devient la resource fork du fichier de structure Macintosh. A l'inverse, lorsque vous transportez une base 4D de MacOS vers Windows, le fichier de structure Macintosh est **dissocié** en deux fichiers distincts. Sa data fork devient le fichier .4DB et sa resource fork devient le fichier .RSR.

La dissociation ou la fusion de la data fork et de la resource fork constituent en fait l'unique rôle de 4D Transporter. Le programme ne traduit ni ne modifie en aucune manière les données stockées dans les fichiers. Pour plus d'informations sur le transport de bases 4D entre les plates-formes, reportez-vous au manuel de référence de 4D Transporter.

Fichiers de ressources

Quelle que soit la plate-forme que vous utilisez, le fichier de structure d'une base 4D n'est pas le seul type de fichier utilisant des ressources. L'application 4D elle-même contient des ressources. Sous MacOS, elles sont stockées dans la resource fork de l'application. Sous Windows, elles sont stockées dans le fichier 4D.RSR — le code exécutable est stocké quant à lui dans le fichier 4D.EXE.

Les plug-ins 4D peuvent également contenir des ressources. Par exemple, le plug-in 4D Write de 4D utilise des ressources. Sous MacOS, elles sont stockées dans la resource fork de 4D Write. Sous Windows, elles sont stockées dans le fichier 4DWRITExx.RSR (varie en fonction du numéro de version).

Le fichier de données d'une base 4D peut également contenir des ressources. Par exemple, à l'aide de l'utilitaire **Customizer Plus** (fourni avec les versions Windows et Macintosh de 4e Dimension), vous pouvez verrouiller un fichier de données de manière à ce qu'il ne puisse être utilisé que par un fichier de structure particulier. Cette opération s'effectue par la création de la même ressource WEDD ("wedding" signifiant "mariage") dans les fichiers de structure et de données. Sous MacOS, la ressource est stockée dans la resource fork du fichier de données. Sous Windows, elle est stockée dans le fichier .4DR, le fichier de ressources du fichier de données.

Sous Windows, exception faite du fichier .4DR pour le fichier de données, vous pourrez généralement reconnaître à leur extension .RSR les fichiers 4D standard contenant des ressources de type MacOS transformées en fichier.

En plus des fichiers de ressources fournis par 4D, vous pouvez créer et utiliser vos propres fichiers de ressources à l'aide des commandes 4D Creer fichier ressources et Ouvrir fichier ressources. Lorsque leur exécution s'est déroulée correctement, ces deux commandes retournent un **numéro de référence de fichier de ressources** identifiant de manière unique le fichier de ressources ouvert. Ce numéro équivaut au numéro de référence de document retourné, pour les fichiers standard, par les commandes du thème Documents système, telles que Ouvrir document. Toutes les commandes 4D de gestion des ressources acceptent un numéro de référence de fichier de ressources (optionnel). Une fois que vous en avez terminé avec un fichier de ressources, n'oubliez pas de le refermer en appelant la commande FERMER FICHIER RESSOURCES.

La chaîne des fichiers de ressources

Lorsque vous travaillez avec une base 4D, vous pouvez soit utiliser **tous les fichiers de ressources ouverts** soit **un fichier de ressources particulier**.

Plusieurs fichiers de ressources peuvent être ouverts simultanément. C'est d'ailleurs toujours le cas lorsqu'une base 4D est en cours d'utilisation :
• Sur Macintosh, le fichier de ressources du système est ouvert.
• Sous Windows, le fichier ASIPORT.RSR est ouvert (il contient une partie des ressources système du Macintosh).

• Le fichier de ressources de l'application 4D est ouvert.
• Le fichier de ressources de la structure de la base est ouvert.
• Le fichier de ressources des données de la base est ouvert (s'il existe).
• Enfin, vous pouvez ouvrir votre propre fichier de ressources à l'aide de la fonction Ouvrir fichier ressources.

Cette liste de ressources ouvertes s'appelle la **chaîne des fichiers de ressources**. Lorsque vous recherchez une ressource particulière, celle-ci peut être désignée de deux manières :
• Si vous passez un numéro de référence de fichier de ressources à une commande 4D de gestion des ressources, la ressource est recherchée dans ce fichier uniquement.
• Si vous ne passez pas de numéro de référence de fichier de ressources à la commande 4D, la ressource est recherchée dans tous les fichiers de ressources ouverts, depuis le plus récemment ouvert jusqu'au premier ouvert. 4D remonte en sens inverse la chaîne des fichiers de ressources ouverts : le dernier fichier ouvert est examiné en premier.

Voici un exemple :

```
$vhResFile:=Creer fichier ressources("Simple_Fichier")
Si (OK=1)
   TABLEAU ALPHA(63;asDesChaines;0)
   LISTE DE CHAINES VERS TABLEAU(8;asDesChaines;$vhResFile)
   ALERTE("Le tableau contient "+Chaine(Taille tableau(asDesChaines))+
                                                          " élément(s).")
   LISTE DE CHAINES VERS TABLEAU(8;asDesChaines)
   ALERTE("Le tableau contient "+Chaine(Taille tableau(asDesChaines))+
                                                          " élément(s).")
   FERMER FICHIER RESSOURCES($vhResFile)
Fin de si
```

Lors de l'exécution de cette méthode, la première alerte affiche “Le tableau contient 0 élément(s)” et la seconde alerte affiche “Le tableau contient 634 élément(s)”.

Le premier appel :

```
LISTE DE CHAINES VERS TABLEAU(8;asDesChaines;$vhResFile)
```

recherche la ressource "STR#" ID=8 uniquement dans le fichier de ressources qui vient d'être créé et ouvert par la commande Creer fichier ressources. Comme ce fichier est neuf et donc vide, la recherche ne donne rien.

Le second appel :

```
LISTE DE CHAINES VERS TABLEAU(8;asDesChaines)
```

recherche la ressource "STR#" ID=8 dans tous les fichiers de ressources ouverts. Comme le fichier qui vient d'être créé et ouvert par la commande Creer fichier ressources ne contient pas cette ressource, LISTE DE CHAINES VERS TABLEAU recherche alors la ressource dans le fichier de ressources de la structure de la base. Ce fichier ne la contenant pas non plus, LISTE DE CHAINES VERS TABLEAU examine alors le fichier de ressources de l'application 4D et y trouve finalement la ressource recherchée. Le tableau est alors rempli.

Conclusion : Lorsque vous travaillez avec des fichiers de ressources, vous devez passer un numéro de référence de fichier de ressources aux commandes 4D de gestion des ressources, si vous voulez accéder à un fichier spécifique. Sinon, 4D considère que vous ne souhaitez pas utiliser de ressources en provenance d'un fichier particulier.

Types de ressources

Le format interne d'un fichier de ressources est très structuré. En plus des données de chaque ressource, le fichier contient un en-tête et un descriptif qui fournissent des informations précises sur son contenu.

Les ressources sont classées en **types**. Le type d'une ressource est indiqué par une chaîne de 4 caractères. Par exemple :
• Une ressource de type "STR#" est une ressource contenant une liste de chaînes Pascal. Elle est appelée **ressource liste de chaînes**.
• Une ressource de type "STR " (notez que le quatrième caractère est un caractère d'espacement) est une ressource contenant une chaîne Pascal individuelle. Elle est appelée **ressource chaîne**.
• Une ressource de type "TEXT" est une ressource contenant du texte sans longueur déclarée. Elle est appelée **ressource texte**.
• Une ressource de type "PICT" est une ressource contenant une image QuickDraw Macintosh que vous pouvez utiliser et afficher sous MacOS et Windows avec 4D. Elle est appelée **ressource image**.
• Une ressource de type "cicn" est une ressource contenant une icône couleur Macintosh que vous pouvez utiliser et afficher sous MacOS et Windows avec 4D. Une ressource "cicn" peut, par exemple, être associée à un élément d'une liste hiérarchique à l'aide de la commande CHANGER PROPRIETES ELEMENT. Elle est appelée **ressource icône couleur**.

En plus des types de ressources standard (la liste ci-dessus n'est pas exhaustive), vous pouvez créer vos propres types. Par exemple, vous pouvez décider de travailler avec des ressources du type "MTYP" (pour “Mon Type”).

Pour obtenir la liste des types de ressources présents parmi tous les fichiers ouverts ou dans un fichier particulier, utilisez la commande LISTE TYPES RESSOURCE. A l'inverse, pour obtenir la liste des ressources d'un certain type parmi tous les fichiers de ressources ouverts ou dans un fichier de ressources particulier, utilisez la commande LISTE RESSOURCES. Cette dernière retourne les numéros et les noms (cf. section suivante) de toutes les ressources d'un type particulier.

Un type de ressource est toujours indiqué par une chaîne de 4 caractères. Les caractères diacritiques et les majuscules/minuscules sont pris en compte. Par exemple, les types de ressources "Hi_!", "hi_!" et "HI_!" sont tous différents.

Important : Les types de ressources en caractères minuscules sont réservés pour le Système d'exploitation. Evitez de désigner vos propres types de ressources en utilisant des caractères minuscules.

ATTENTION : De nombreuses applications s'appuient sur le type des ressources pour traiter leur contenu. Par exemple, lorsqu'elles accèdent à une ressource "STR#", les applications s'attendent à trouver une liste de chaînes. Ne stockez pas de données atypiques dans des ressources de type standard, cela peut provoquer des erreurs système dans vos applications 4D ou dans d'autres applications.

ATTENTION : Comme un fichier de ressources est très structuré, vous ne devez pas y accéder par des commandes autres que celles de gestion des ressources. Notez que si vous passez un numéro de référence de fichier de ressources (sous forme d'une expression 4D de type heure, tout comme pour les numéros de référence de document) à une commande telle que ENVOYER PAQUET, rien ne vous en empêchera (aucune mise en garde ne vous est adressée). Mais il est très probable que le fichier de ressources soit endommagé par l'opération.

ATTENTION : Un fichier de ressources peut contenir jusqu'à 2 700 ressources individuelles. Prenez garde à ne pas de dépasser cette limite (aucune mise en garde ne vous est adressée, mais le fichier de ressources devient endommagé et inutilisable).

Nom et numéro de ressource

Toute ressource a un **nom de ressource**. Un nom de ressource peut contenir jusqu'à 255 caractères, tient compte des caractères diacritiques mais n'établit pas de distinction entre les majuscules et les minuscules. Les noms des ressources peuvent être utiles pour leur identification visuelle, mais généralement vous accèderez à une ressource par l'intermédiaire de son numéro. Les noms des ressources ne sont pas uniques, plusieurs ressources peuvent avoir le même nom.

Toute ressource a un **numéro d'identification** (on dit aussi numéro d'ID ou ID). Ce numéro d'ID est unique à l'intérieur d'un type et d'un fichier de ressources. Par exemple :
• Un fichier de ressource peut contenir une ressource "ABCD" ID=1 et une ressource "EFGH" ID=1.
• Deux fichiers de ressources peuvent contenir une ressource de même type et de même numéro.

Lorsque vous accédez à une ressource par l'intermédiaire d'une commande 4D, vous indiquez son type et son numéro. Si vous ne spécifiez pas le fichier de ressources dans lequel vous souhaitez la rechercher, la commande retournera l'occurrence de la ressource trouvée dans le premier fichier de ressource examiné. Rappelez-vous que les fichiers de ressources sont examinés dans l'ordre inverse de celui dans lequel ils ont été ouverts.

Les numéros de ressources sont compris entre –32 768 et 32 767.

Important : N'utilisez pas de numéros de ressources négatifs, ils sont réservés au Système d'exploitation. N'utilisez pas non plus de numéros situés entre 0 et 14 999, cet intervalle est réservé à 4e Dimension. Pour vos propres ressources, utilisez les numéros situés entre 15 000 et 32 767.

Pour obtenir les numéros et les noms de ressources d'un type particulier, utilisez la commande LISTE RESSOURCES.
Pour obtenir le nom d'une ressource individuelle, utilisez la commande Lire nom ressource.
Pour changer le nom d'une ressource individuelle, utilisez la commande ECRIRE NOM RESSOURCE.
Pour obtenir le numéro courant d'une ressource PICT ou STR# installée par un composant 4D, utilisez la commande Lire ID ressource composant.

Comme chaque commande 4D accepte de manière optionnelle un numéro de référence de fichier de ressources, vous pouvez facilement manipuler des ressources ayant le même type et le même numéro mais situées dans deux fichiers de ressources différents.
L'exemple suivant copie toutes les ressources "PICT" d'un fichier de ressources dans un autre :

```
    ` Ouverture d'un fichier de ressources existant
$vhResFileA:=Ouvrir fichier ressources("")
Si (OK=1)
        ` Création d'un nouveau fichier de ressources
   $vhResFileB:=Creer fichier ressources("")
   Si (OK=1)
        ` Récupérer la liste des numéros et des noms de toutes les ressources de type
        ` "PICT" situées dans le fichier de ressources A
      LISTE RESSOURCES("PICT";$aiResID;$asResName;$vhResFileA)
        ` Pour chaque ressource :
      Boucle($vlElem;1;Taille tableau($aiResID))
         $viResID:=$aiResID{$vlElem}
            ` Charger la ressource du fichier A
         LIRE RESSOURCE ("PICT";$viResID;vxResData;$vhResFileA)
            ` Si la ressource peut être chargée
         Si (OK=1)
               ` Ecrire la ressource dans le fichier B
            ECRIRE RESSOURCE ("PICT";$viResID;vxResData;$vhResFileB)
               ` Si la ressource peut être écrite
            Si (OK=1)
                  ` Copie également du nom de la ressource...
               ECRIRE NOM RESSOURCE("PICT;$viResID;$asResName{$vlElem}
                                                              ;$vhResFileB)
                  ` ...Ainsi que ses propriétés (cf. § ci-dessous)
               $vlResAttr:=Lire proprietes ressource("PICT";$viResID;$vhResFileA)
               ECRIRE PROPRIETES RESSOURCE("PICT";$viResID;$vlResAttr;
                                                              $vhResFileB)
            Sinon
               ALERTE("La ressource PICT ID="+Chaine($viResID)+" ne peut pas être
                                                                  créée.")
            Fin de si
         Sinon
            ALERTE("La ressource PICT ID="+Chaine($viResID)+" ne peut pas être
                                                                  chargée.")
         Fin de si
```

```
        Fin de boucle
        FERMER FICHIER RESSOURCES($vhResFileB)
    Fin de si
    FERMER FICHIER RESSOURCES($vhResFileA)
Fin de si
```

Propriétés des ressources

En plus de son type, de son nom et de son numéro, une ressource possède des **propriétés** supplémentaires (aussi appelées attributs). Par exemple, une ressource peut être purgeable ou non. Cet attribut indique au Système d'exploitation si, une fois la ressource chargée en mémoire, il peut ou non la purger (c'est-à-dire l'effacer) en cas de besoin de mémoire supplémentaire. Comme le montre l'exemple précédent, il peut être important lors de la copie ou de la création d'une ressource de ne pas copier uniquement la ressource, mais également son nom et ses propriétés. Pour plus d'informations sur les propriétés des ressources, reportez-vous aux descriptions des commandes Lire proprietes ressource et ECRIRE PROPRIETES RESSOURCE.

Manipuler le contenu des ressources

Pour charger en mémoire tout type de ressource, utilisez la commande LIRE RESSOURCE, qui retourne le contenu de la ressource dans un BLOB. Pour créer ou réécrire une ressource sur disque, appelez la commande ECRIRE RESSOURCE, qui utilise le contenu du BLOB que vous passez en paramètre pour écrire le contenu de la ressource. Pour supprimer une ressource existante, utilisez la commande SUPPRIMER RESSOURCE.

Pour simplifier la manipulation des ressources, 4D dispose de commandes intégrées supplémentaires dédiées à la gestion des ressources de type standard. Ces commandes vous évitent de devoir analyser des BLOBs pour pouvoir en extraire le contenu des ressources qui vous intéressent :

- LISTE DE CHAINES VERS TABLEAU remplit un tableau Alpha ou Texte avec les chaînes de caractères contenues dans une ressource liste de chaînes.
- TABLEAU VERS LISTE DE CHAINES crée ou réécrit une ressource liste de chaînes avec les éléments d'un tableau Alpha ou Texte.
- Lire chaine dans liste retourne une chaîne particulière d'une ressource liste de chaînes.
- Lire ressource chaine retourne la chaîne d'une ressource chaîne.
- ECRIRE RESSOURCE CHAINE crée ou réécrit une ressource chaîne.
- Lire ressource texte retourne le texte d'une ressource texte.
- ECRIRE RESSOURCE TEXTE crée ou réécrit une ressource texte.
- LIRE RESSOURCE IMAGE retourne l'image d'une ressource image.
- ECRIRE RESSOURCE IMAGE crée ou réécrit une ressource image.
- LIRE RESSOURCE ICONE retourne une icône couleur en tant qu'image.

Notez que ces commandes sont fournies afin de simplifier l'emploi des ressources de type standard, mais vous pouvez parfaitement utiliser LIRE RESSOURCE et ECRIRE RESSOURCE avec des BLOBs. Par exemple, la ligne de code suivante :

```
ALERTE(Lire ressource texte(20000))
```

est équivalente (en plus court) à :

```
LIRE RESSOURCE("TEXT";20000;vxData)
Si (OK=1)
   $vlOffset:=0
   ALERTE(BLOB vers texte(vxData;Texte sans longueur;$vlOffset;
                                                    Taille BLOB(vxData)))
Fin de si
```

Les commandes 4D et les ressources

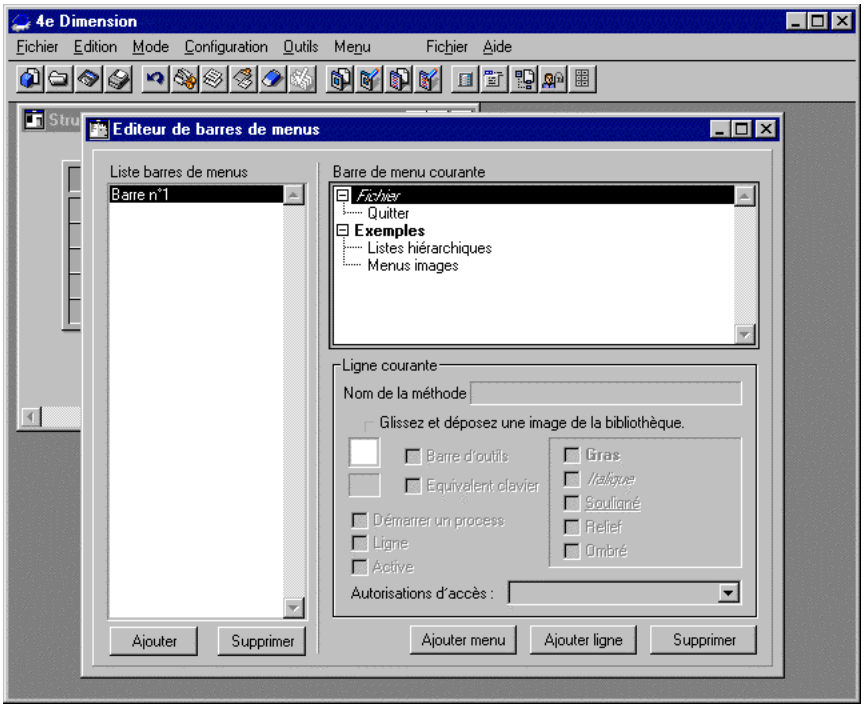
Outre les commandes de gestion des ressources décrites dans ce chapitre, plusieurs commandes 4D vous permettent de travailler avec des fichiers de ressources :
• Sur Macintosh, DOCUMENT VERS BLOB et BLOB VERS DOCUMENT peuvent charger et écrire la totalité de la resource fork d'un fichier Macintosh.
• A l'aide des commandes CHANGER PROPRIETES ELEMENT et CHANGER PROPRIETES LISTE, vous pouvez associer des ressources images ou icônes couleur aux éléments d'une liste, ou encore utiliser des ressources icônes couleur en tant qu'icônes des éléments parents d'une liste hiérarchique (déployés/contractés).
• La commande JOUER SON joue des ressources "snd " (sous MacOS et Windows).
• La commande CHANGER POINTEUR SOURIS peut utiliser des ressources "CURS" pour modifier l'apparence du pointeur de la souris.

Référence

Commandes du thème BLOB, Erreurs du gestionnaire de ressources du système, Les ressources et 4D Insider : un exemple, Lire ID ressource composant.

Les ressources et 4D Insider : un exemple

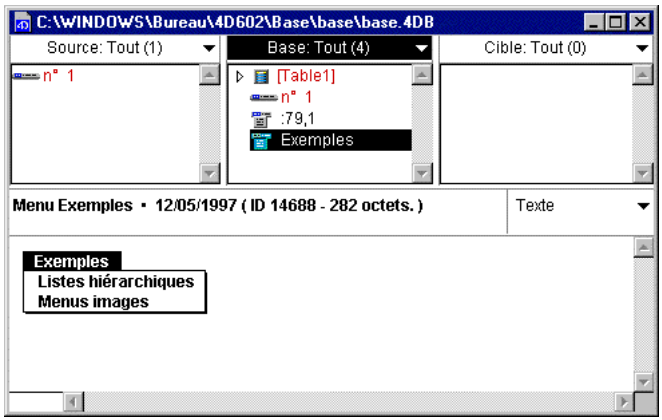
Les ressources sont un moyen très pratique de traiter les questions de localisation liées au développement et à la maintenance de bases 4D destinées au marché international, et devant être disponibles en plusieurs langues. Nous allons examiner un exemple. L'écran suivant montre la barre de menus personnalisée d'une base en français :



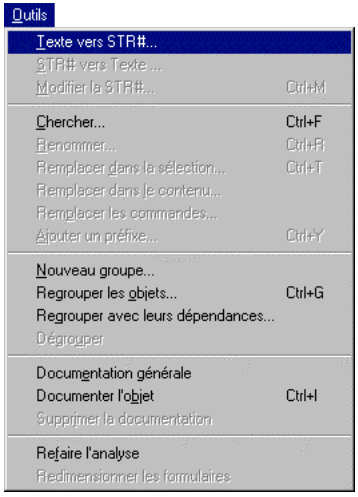
Le libellé du menu **Fichier** référence déjà une ressource, mais pas la ligne **Quitter**. Le menu **Exemples** se compose des lignes **Listes hiérarchiques** et **Menus images**. Ni le libellé du menu ni ses lignes ne référencent de ressources.
A l'aide de 4D Insider, vous pouvez transformer les libellés de la barre de menus en références à des chaînes de caractères stockées dans des ressources STR#. Nous allons voir comment réaliser cette opération.

Note : 4D Insider est l'outil de gestion des références croisées et des librairies d'objets de 4D. Il est compris dans l'offre 4D Desktop.

Ouvrez la base de données avec 4D Insider. L'écran suivant présente la barre de menus dans la fenêtre de navigation de 4D Insider :

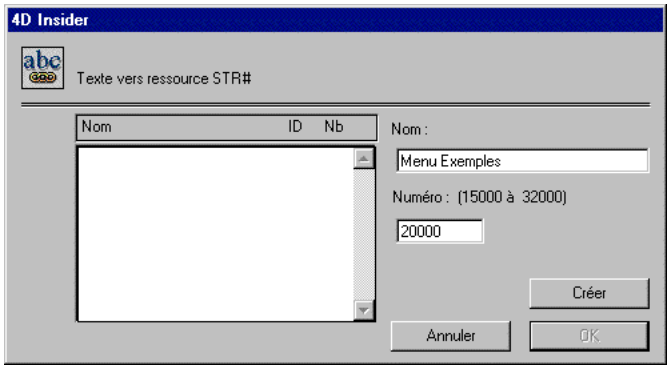


Vous pouvez alors convertir les libellés de la barre de menus afin qu'ils référencent une ressource STR#. Pour cela, choisissez la commande **Texte vers STR#** dans le menu **Outils** de 4D Insider :

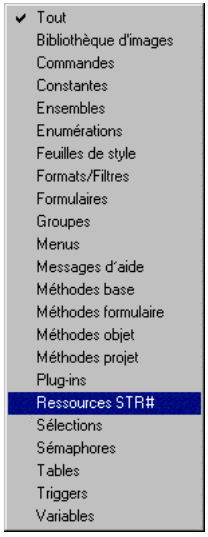


La boîte de dialogue **Texte vers ressource STR#** apparaît.

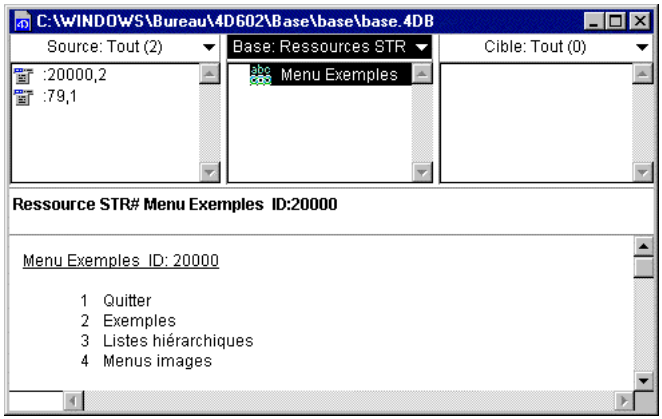
Saisissez, par exemple, **Menu Exemples** et **20000** comme nom et numéro de ressource :



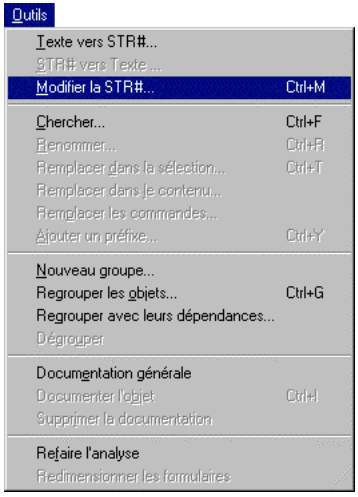
Cliquez sur le bouton **Créer**, puis sur **OK**.
Sélectionnez **Ressources STR#** dans le pop up menu de la liste Base de la fenêtre de navigation :



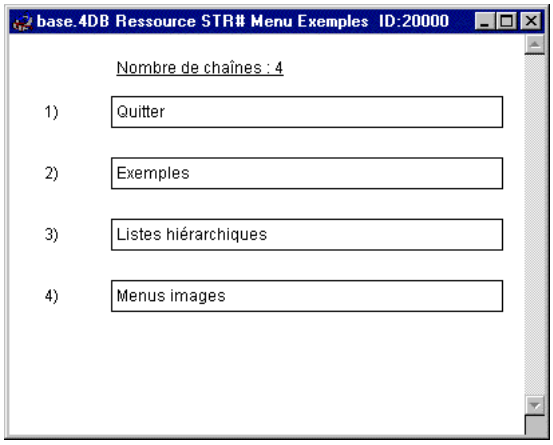
Double-cliquez sur la STR# **Menu Exemples** afin d'afficher son contenu :



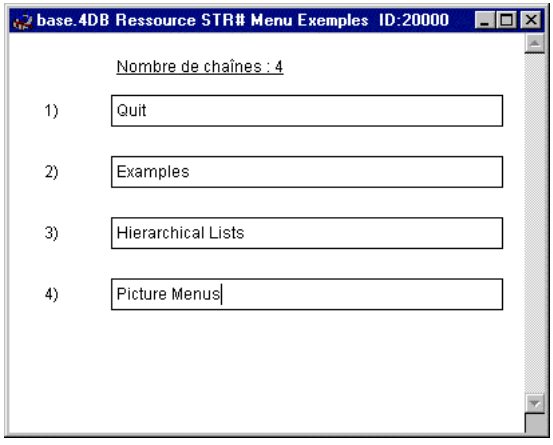
Maintenant que ces chaînes sont stockées dans une ressource, vous pouvez les modifier sans perturber la logique de développement de votre base. Pour cela, assurez-vous que l'élément **Menu Exemples** est bien sélectionné dans la liste Base, puis choisissez la commande **Modifier la STR#** dans le menu **Outils** de 4D Insider :



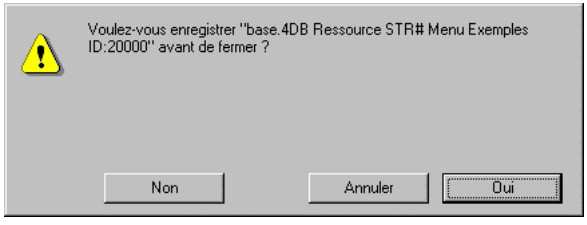
L'éditeur de ressources STR# de 4D Insider apparaît :



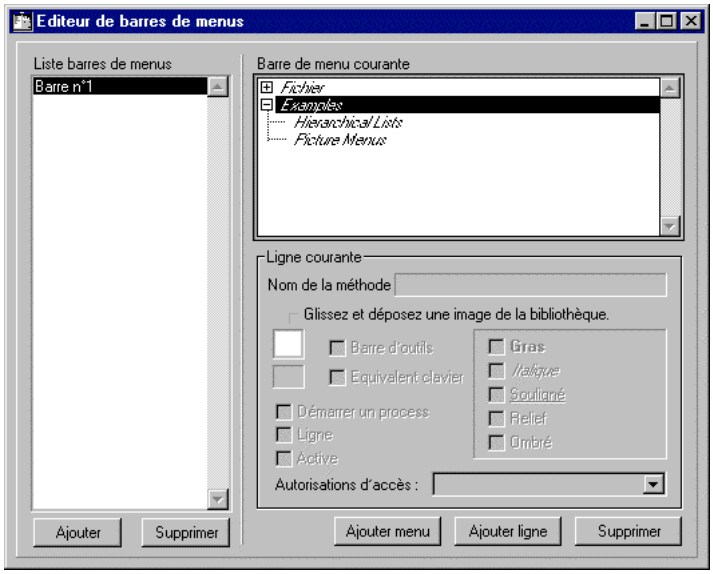
Traduisez les chaînes de caractères dans une autre langue, par exemple en anglais :



Une fois que vous avez terminé la traduction, fermez la fenêtre. Cliquez sur **Oui** dans la boîte de dialogue de confirmation :

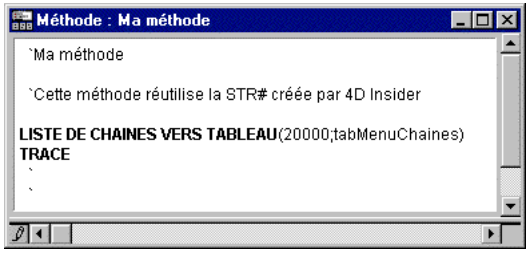


Quittez ensuite 4D Insider et réouvrez la base avec 4e Dimension. L'éditeur de barres de menus de 4D (en mode Structure) affiche désormais la barre n°1 avec les références aux ressources en anglais :

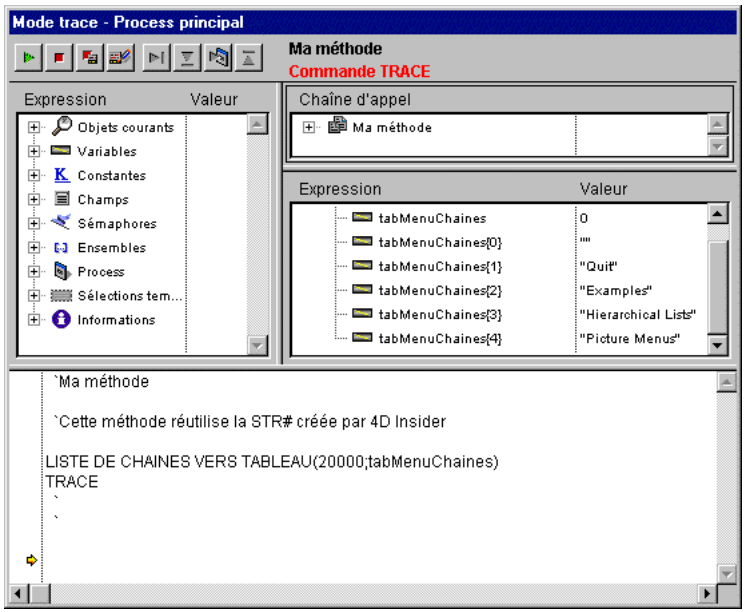


Pour plus d'informations sur cette opération, reportez-vous à la documentation 4D Insider, si vous avez souscrit à l'offre 4D Desktop. Plus généralement, pour plus d'informations sur l'utilisation de références à des ressources dans les barres de menus ainsi que dans les objets des formulaires de vos bases 4D, reportez-vous au manuel *Mode Structure* de 4e Dimension.

Les commandes 4D du thème Ressources peuvent exploiter les ressources créées par 4D Insider. La méthode présentée ci-dessous utilise la commande LISTE DE CHAINES VERS TABLEAU pour charger dans un tableau la ressource STR# resource créée à l'aide de 4D Insider :



Dans la fenêtre du débogueur, vous pouvez constater que le tableau est rempli avec les chaînes traduites dans 4D Insider :



Référence

Ressources.

Ouvrir fichier ressources

Ressources

version 6.0

Ouvrir fichier ressources (resNomFichier{; typeFichier}) → DocRef

Paramètre	Type		Description
resNomFichier	Alpha	→	Nom ou chemin d'accès complet du fichier de ressources ou chaîne vide pour afficher la boîte de dialogue standard d'ouverture de fichiers
typeFichier	Alpha	→	Type de fichier MacOS (chaîne de 4 caractères), ou extension de fichier Windows (chaîne de 1 à 3 caractères), ou fichier de ressources ("res " / .RES) si omis
Résultat	DocRef	←	Numéro de référence du fichier de ressources

Description

La commande Ouvrir fichier ressources ouvre le fichier de ressources dont vous avez passé le nom ou le chemin d'accès complet dans le paramètre resNomFichier.

Si vous passez un nom de fichier, celui-ci doit se trouver dans le même dossier/répertoire que le fichier de structure de la base. Pour ouvrir un fichier de ressources se trouvant dans un autre dossier, passez un chemin d'accès complet dans resNomFichier.

Si vous passez une chaîne vide dans resNomFichier, la boîte de dialogue standard d'ouverture de fichiers apparaît, permettant à l'utilisateur de sélectionner le fichier à ouvrir. Si l'utilisateur clique sur **Annuler** dans cette boîte de dialogue, aucun fichier de ressources n'est ouvert, Ouvrir fichier ressources retourne une valeur nulle dans RefDoc et la variable OK prend la valeur 0.

Si le fichier de ressources est correctement ouvert, Ouvrir fichier ressources retourne son numéro de référence de fichier et met la variable OK à 1. Si le fichier de ressources n'existe pas ou si le fichier de que vous tentez d'ouvrir n'est pas un fichier de ressources, une erreur est générée.

Sous MacOS, si vous utilisez la boîte de dialogue standard d'ouverture de fichiers, tous les fichiers sont présentés par défaut. Pour ne faire apparaître que les fichiers d'un type particulier, spécifiez-le dans le paramètre optionnel typeFichier.

Sous Windows, si vous utilisez la boîte de dialogue standard d'ouverture de fichiers, tous les fichiers sont présentés par défaut. Pour ne faire apparaître que les fichiers d'un type particulier, passez dans typeFichier une extension de fichier Windows de 1 à 3 caractères ou un type de fichier Macintosh associé à une extension Windows à l'aide de la commande ASSOCIER TYPES FICHIER.

N'oubliez pas d'appeler finalement FERMER FICHIER RESSOURCES pour le fichier de ressources. Notez cependant que 4D referme automatiquement tous les fichiers de ressources ouverts par l'intermédiaire de Ouvrir fichier ressources ou Creer fichier ressources lorsque vous quittez l'application ou ouvrez une autre base de données.

A la différence de la commande Ouvrir document qui ouvre un document (la *data fork* sous MacOS) avec un accès exclusif en lecture-écriture, Ouvrir fichier ressources vous permet d'ouvrir un fichier de ressources déjà ouvert dans la session 4D. Par exemple, lorsque vous tentez d'ouvrir deux fois le même document avec Ouvrir document, une erreur d'E/S vous est retournée lors de la seconde opération. En revanche, vous pouvez accéder à un fichier de ressources déjà ouvert lors de la session 4D : Ouvrir fichier ressources retourne son numéro de référence. Même lorsque vous ouvrez plusieurs fois un fichier de ressources, il vous suffit d'appeler FERMER FICHIER RESSOURCES une seule fois pour refermer ce fichier. Notez que ce fonctionnement n'est valable que lorsque le fichier de ressources est ouvert à l'intérieur de la session 4D. Si vous tentez d'ouvrir un fichier de ressources déjà ouvert par une autre application, une erreur d'E/S vous sera retournée.
Grâce à ces possibilités d'ouvertures multiples, vous pouvez facilement obtenir les numéros de référence des fichiers de ressources de la base ou de l'application 4D sans interférer avec le déroulement normal des opérations dans 4D (cf. exemples 5 et 6).

ATTENTION

• Soyez très prudent si vous accédez au fichier de ressources de l'application 4D. Ne modifiez en aucun cas ces ressources, vous pourriez par inadvertance endommager le programme et provoquer des erreurs système. Rappelez-vous également que votre base peut être utilisée dans des environnements différents (4D, 4D Runtime, 4D Engine, 4D Server et 4D Client).
• Si vous accédez au fichier de ressources de la base et souhaitez ajouter, supprimer ou modifier des ressources par programmation, pensez à tester l'environnement dans lequel la base s'exécute. Avec 4D Server, cela posera certainement d'épineux problèmes. Si, par exemple, vous modifiez une ressource sur le poste serveur (via une méthode base ou une procédure stockée), vous allez en définitive perturber le système d'administration de 4D Server chargé de distribuer de manière transparente les ressources aux postes clients. Notez qu'avec 4D Client vous n'accédez pas directement au fichier de structure : il est situé sur le poste serveur.
• Pour toutes ces raisons, si vous exploitez des ressources, nous vous conseillons de les stocker dans vos propres fichiers.
• Lorsque vous travaillez avec vos propres ressources, n'utilisez pas de numéros de ressources négatifs, ils sont réservés au Système d'exploitation. N'utilisez pas non plus de numéros situés entre 0 et 14 999, cet intervalle est réservé à 4e Dimension. Pour vos propres ressources, utilisez les numéros situés entre 15 000 et 32 767. Rappelez-vous que dès qu'un fichier de ressources est ouvert, il devient le premier maillon de la chaîne des fichiers de ressources, et c'est dans ce fichier que les ressources seront recherchées en premier lieu. En conséquence, si vous stockez dans ce fichier des ressources dont les numéros appartiennent aux intervalles réservés au Système ou à 4D, ces ressources seront utilisées non seulement par les commandes telles que LIRE RESSOURCE mais également par les routines internes de l'application 4D elle-même. Si vous n'êtes pas absolument certain de ce que vous faites, n'utilisez pas les intervalles réservés, cela peut conduire à des erreurs système.

• Un fichier de ressources est très structuré et ne peut contenir plus 2 700 ressources. Si vous travaillez avec des fichiers comportant un grand nombre de ressources, il est conseillé de tester ce nombre avant d'ajouter de nouvelles ressources à un fichier (reportez-vous à l'exemple Nombre de ressources dans la description de la commande LISTE TYPES RESSOURCE).

Une fois que vous avez ouvert un fichier de ressources, vous pouvez analyser son contenu à l'aide des commandes LISTE TYPES RESSOURCE et LISTE RESSOURCES.

Exemples

(1) Dans l'exemple suivant, nous cherchons à ouvrir sous Windows le fichier de ressources "MesPrefs.res" situé dans le dossier de la base :

```
⇒ $vhResFile:=Ouvrir fichier ressources("MesPrefs";"res ")
```

Sous MacOS, l'exemple recherchera le fichier “MesPrefs”.

(2) Cet exemple tente d'ouvrir sous Windows le fichier de ressources "MesPrefs.rsr" situé dans le dossier de la base :

```
⇒ $vhResFile:=Ouvrir fichier ressources("MesPrefs";"rsr")
```

Sous MacOS, l'exemple recherchera le fichier “MesPrefs”.

(3) L'exemple suivant fait apparaître la boîte de dialogue standard d'ouverture de fichiers, dans laquelle tous les types de documents sont affichés :

```
⇒ $vhResFile:=Ouvrir fichier ressources("")
```

(4) L'exemple suivant fait apparaître la boîte de dialogue standard d'ouverture de fichiers, dans laquelle seuls les documents créés à l'aide de la fonction Creer fichier ressources et utilisant le type par défaut sont affichés :

```
⇒ $vhResFile:=Ouvrir fichier ressources("";"res ")
   Si (OK=1)
      ALERTE("Vous venez d'ouvrir ""+Document+""".")
      FERMER FICHIER RESSOURCES($vhResFile)
   Fin de si
```

(5) L'exemple suivant retourne dans $vhStructureResFile le numéro de référence du fichier de ressources de la structure de la base :

```
   Si (Sous Windows)
⇒     $vhStructureResFile:=Ouvrir fichier ressources(Remplacer chaine(Fichier
                                                    structure;".4DB";".RSR"))
   Sinon
⇒     $vhStructureResFile:=Ouvrir fichier ressources(Fichier structure)
   Fin de si
```

(6) L'exemple suivant retourne dans $vhApplResFile le numéro de référence du fichier de ressources de l'application 4D :

```
      Si (Sous Windows)
⇒        $vhApplResFile:=Ouvrir fichier ressources(Remplacer chaine(Fichier
                                                                application;".EXE";".RSR"))
      Sinon
⇒        $vhApplResFile:=Ouvrir fichier ressources(Fichier application)
      Fin de si
```

Référence

Creer fichier ressources, FERMER FICHIER RESSOURCES, Ressources.

Variables et ensembles système

Si le fichier de ressources est correctement ouvert, la variable système OK prend la valeur 1. Si le fichier de ressources n'a pas pu être ouvert ou si l'utilisateur a cliqué sur le bouton Annuler dans la boîte de dialogue standard d'ouverture de fichiers, la variable OK prend la valeur 0 (zéro).

Si le fichier de ressources est correctement ouvert par l'intermédiaire de la boîte de dialogue standard d'ouverture de fichiers, la variable système Document contient le chemin d'accès au fichier.

Gestion des erreurs

Si le fichier de ressources n'a pas pu être ouvert à la suite d'un problème de ressource ou d'E/S, une erreur est générée. Vous pouvez intercepter cette erreur à l'aide d'une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Creer fichier ressources

Ressources

version 6.0

Creer fichier ressources (resNomFichier{; typeFichier}) → DocRef

Paramètre	Type		Description
resNomFichier	Alpha	→	Nom ou chemin d'accès complet du fichier de ressources ou chaîne vide pour afficher la boîte de dialogue standard d'enregistrement de fichiers
typeFichier	Alpha	→	Type de fichier MacOS (chaîne de 4 caractères), ou extension de fichier Windows (chaîne de 1 à 3 caractères), ou fichier de ressources ("res " / .RES) si omis
Résultat	DocRef	←	Numéro de référence du fichier de ressources

Description

La commande Creer fichier ressources crée et ouvre un nouveau fichier de ressources à partir du nom ou du chemin d'accès complet que vous avez passé dans resNomFichier.

Si vous passez un nom de fichier, celui-ci sera placé dans le même dossier que le fichier de structure de la base. Passez un chemin d'accès complet pour créer un fichier de ressources dans un autre dossier.

Si le fichier existe déjà et n'est pas ouvert, Creer fichier ressources le remplace par le nouveau fichier de ressources vide. Si le fichier existant est ouvert, une erreur d'E/S est retournée.

Si vous passez une chaîne vide dans resNomFichier, la boîte de dialogue standard d'enregistrement de fichiers apparaît, permettant à l'utilisateur de choisir le nom et l'emplacement du fichier de ressources à créer. Si l'utilisateur clique sur le bouton **Annuler** dans la boîte de dialogue, aucun fichier de ressources n'est créé, Creer fichier ressources retourne une valeur nulle dans DocRef et la variable système OK prend la valeur 0.

Si le fichier de ressources est correctement créé et ouvert, Creer fichier ressources retourne son numéro de référence de fichier de ressources et la variable OK prend la valeur 1. Si le fichier de ressources ne peut pas être créé, une erreur est générée.

Sous MacOS, le type par défaut d'un fichier créé avec Creer fichier ressources est "res ". Sous Windows, l'extension de fichier par défaut est ".res". Si vous voulez créer un fichier d'un autre type :
• Sous MacOS, passez le type du fichier dans le paramètre optionnel typeFichier.
• Sous Windows, passez dans typeFichier une extension Windows de 1 à 3 caractères ou un type de fichier Macintosh associé à l'aide de la commande ASSOCIER TYPES FICHIER.

N'oubliez pas d'appeler finalement FERMER FICHIER RESSOURCES pour le fichier de ressources. Notez cependant que 4D referme automatiquement tous les fichiers de ressources ouverts par l'intermédiaire de Creer fichier ressources ou Ouvrir fichier ressources lorsque vous quittez l'application ou ouvrez une autre base de données.

Exemples

(1) L'exemple suivant crée et ouvre sous Windows le fichier de ressources "MesPrefs.res" dans le dossier de la base :

```
⇒ $vhResFile:=Creer fichier ressources("MesPrefs")
```

Sous MacOS, l'exemple crée et ouvre le fichier “MesPrefs”.

(2) L'exemple suivant crée et ouvre sous Windows le fichier de ressources "MesPrefs.rsr" dans le dossier de la base :

```
⇒ $vhResFile:=Creer fichier ressources("MesPrefs";"rsr")
```

Sous MacOS, l'exemple crée et ouvre le fichier “MesPrefs”.

(3) L'exemple suivant affiche la boîte de dialogue standard d'enregistrement de fichiers :

```
⇒ $vhResFile:=Creer fichier ressources("")
   Si (OK=1)
      ALERTE("Vous venez de créer “"+Document+"”.")
      FERMER FICHIER RESSOURCES($vhResFile)
   Fin de si
```

Référence

APPELER SUR ERREUR, FERMER FICHIER RESSOURCES, Ouvrir fichier ressources, Ressources.

Variables et ensembles système

Si le fichier de ressources est correctement créé et ouvert, la variable système OK prend la valeur 1. Si le fichier de ressources n'a pas pu être ouvert ou si l'utilisateur a cliqué sur le bouton **Annuler** dans la boîte de dialogue standard d'enregistrement de fichiers, la variable OK prend la valeur 0 (zéro).

Si le fichier de ressources est correctement créé et ouvert par l'intermédiaire de la boîte de dialogue standard d'enregistrement de fichiers, la variable système Document contient le chemin d'accès au fichier.

Gestion des erreurs

Si le fichier de ressources n'a pas pu être créé ou ouvert à la suite d'un problème de ressource ou d'E/S, une erreur est générée. Vous pouvez intercepter cette erreur à l'aide d'une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

FERMER FICHIER RESSOURCES (resFichier)

Paramètre	Type		Description
resFichier	DocRef	→	Numéro de référence de fichier de ressources

Description

La commande FERMER FICHIER RESSOURCES referme le fichier de ressources dont vous avez passé le numéro de référence dans resFichier.

Même si vous avez ouvert plusieurs fois un fichier de ressources, il vous suffit d'appeler FERMER FICHIER RESSOURCES une seule fois pour le refermer.

Si vous appliquez FERMER FICHIER RESSOURCES au fichier de ressources de l'application 4D ou de la base, la commande le détecte et ne fait rien.

Si vous passez un numéro de référence de fichier de ressources non valide, la commande ne fait rien.

N'oubliez pas d'appeler finalement FERMER FICHIER RESSOURCES pour un fichier de ressources que vous avez ouvert à l'aide des commandes Ouvrir fichier ressources ou Creer fichier ressources. Notez cependant que 4D referme automatiquement tous les fichiers de ressources ouverts lorsque vous quittez l'application ou ouvrez une autre base de données.

Exemple

L'exemple suivant crée un fichier de ressources, ajoute une ressource de type chaîne puis referme le fichier :

```
$vhDocRef:=Creer fichier ressources("Un simple fichier")
Si (OK=1)
   ECRIRE RESSOURCE CHAINE(20000;"Une simple chaîne";$vhDocRef)
⇒  FERMER FICHIER RESSOURCES($vhDocRef)
Fin de si
```

Référence

Creer fichier ressources, Ouvrir fichier ressources.

Variable et ensembles système

Aucun(e) n'est affecté(e).

LISTE TYPES RESSOURCE (resTypes{; resFichier})

Paramètre	Type		Description
resTypes	Tableau alpha	←	Liste des types de ressources disponibles
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou tous les fichiers de ressources ouverts (si ce paramètre est omis)

Description

La commande LISTE TYPES RESSOURCE remplit le tableau resTypes avec les types des ressources présentes dans le(s) fichier(s) de resources ouvert(s).

Si vous passez un numéro de référence de fichier de ressources valide dans le paramètre optionnel resFichier, seules les ressources présentes dans ce fichier seront listées. Si vous ne passez pas le paramètre resFichier, toutes les ressources de tous les fichiers de ressources ouverts seront listées.

Si vous ne pré-déclarez pas le tableau resTypes avant d'appeler LISTE TYPES RESSOURCE, la commande créera par défaut un tableau de type Texte. Si vous pré-déclarez le tableau, vous pouvez lui attribuer le type Alpha ou Texte.

Après l'appel, vous pouvez tester le nombre de types de ressources différents qui ont été trouvés en appliquant la commande Taille tableau au tableau resTypes.

Exemples

(1) L'exemple suivant remplit le tableau atResType avec les types de ressources présents dans tous les fichiers de ressource ouverts :

```
⇒ LISTE TYPES RESSOURCE(atResType)
```

(2) L'exemple suivant vous indique si le fichier de structure MacOS que vous utilisez contient des plug-ins 4D "ancien modèle", qui devront être mis à jour si vous voulez exploiter la base sous Windows :

```
   $vhResFile:=Ouvrir fichier ressources(Fichier structure)
⇒  LISTE TYPES RESSOURCE(atResType;$vhResFile)
   Si (Chercher dans tableau(atResType;"4DEX")>0)
      ALERTE("Cette base contient des plug-ins 4D basés sur l'ancien système."+
                                    (Caractere(13)*2)+"Vous devrez les mettre à jour
                                          pour pouvoir utiliser la base sous Windows.")
   Fin de si
```

Note : Le fichier de structure n'est pas le seul fichier dans lequel des plug-ins "ancien modèle" ont pu être installés. La base peut également être associée à un fichier "Routines Externes" ou "Proc.Ext".

(3) La méthode projet suivante retourne le nombre de ressources présentes dans un fichier de ressources :

```
      ` Méthode projet Compter ressources
      ` Compter ressources ( Heure ) -> Entier long
      ` Compter ressources ( DocRef ) -> Nombre de ressources

   C_ENTIER LONG($0)
   C_HEURE($1)

   $0:=0
⇒  LISTE TYPES RESSOURCE($atResType;$1)
   Boucle ($vlElem;1;Taille tableau($atResType))
      LISTE RESSOURCES($atResType{$vlElem};$alResID;$atResName;$1)
      $0:=$0+Taille tableau($alResID)
   Fin de boucle
```

Une fois que cette méthode est implémentée dans votre base, vous pouvez écrire par exemple :

```
$vhResFile:=Ouvrir fichier ressources("")
Si (OK=1)
   ALERTE("Le fichier “"+Document+"” contient "+Chaine(Compter ressources
                                                ($vhResFile))+" ressource(s).")
   FERMER FICHIER RESSOURCES($vhResFile)
Fin de si
```

Référence

LISTE RESSOURCES.

LISTE RESSOURCES (resType; resNums; resNoms{; resFichier})

Paramètre	Type		Description
resType	Alpha	→	Type de ressource (4 caractères)
resNums	Tableau E. long	←	Numéros des ressources de ce type
resNoms	Tableau alpha	←	Noms des ressources de ce type
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou tous les fichiers de ressources ouverts si ce paramètre est omis

Description

La commande LISTE RESSOURCES remplit les tableaux resNums et resNoms avec les numéros et les noms des ressources dont vous avez passé le type dans resType.

Important : Vous devez passez dans resType une chaîne de 4 caractères.

Si vous passez un numéro de référence de fichier de ressources valide dans le paramètre optionnel resFichier, seules les ressources présentes dans ce fichier seront listées. Si vous ne passez pas le paramètre resFichier, toutes les ressources de tous les fichiers de ressources ouverts seront listées.

Si vous ne pré-déclarez pas les tableaux resNums et resNoms avant d'appeler LISTE RESSOURCES, la commande créera par défaut le tableau resNums avec le type Entier long et resNoms avec le type Texte. Si vous pré-déclarez les tableaux, vous devez attribuer le type Entier long à resNums, mais pouvez attribuer le type Alpha ou Texte à resNoms.

Après l'appel, vous pouvez tester le nombre de ressources qui ont été trouvées en appliquant la commande Taille tableau au tableau resNums ou resNoms.

Exemples

(1) L'exemple suivant remplit les tableaux $alResNum et $atResNom avec les numéros et les noms des ressources de type Listes de chaînes présentes dans le fichier de structure de la base :

```
   Si (Sous Windows)
      $vhStructureResFile:=Ouvrir fichier ressources(Remplacer chaine(Fichier
                                                                  structure;".4DB";".RSR"))
   Sinon
      $vhStructureResFile:=Ouvrir fichier ressources(Fichier structure)
   Fin de si
   Si (OK=1)
⇒     LISTE RESSOURCES("STR#";$alResNum;$atResNom;$vhStructureResFile)
   Fin de si
```

(2) L'exemple suivant copie dans la bibliothèque d'images de la base les ressources image présentes dans tous les fichiers de ressources ouverts :

```
⇒   LISTE RESSOURCES("PICT";$alResNum;$atResNom)
    Creer fenetre(50;50;550;120;5;"Copie des ressources PICT...")
    Boucle ($vlElem;1;Taille tableau($alResNum))
       LIRE RESSOURCE IMAGE($alResNum{$vlElem};$vgImage)
       Si (OK=1)
          $vsNom:=$atResNom{$vlElem}
          Si ($vsNom="")
             $vsNom:="PICT resID="+Chaine($alResNum{$vlElem})
          Fin de si
          EFFACER FENETRE
          POSITION MESSAGE(2;1)
          MESSAGE("Ajout de l'image ""+$vsNom+"" à la bibliothèque d'images de
                                                                    la base.")
          ECRIRE IMAGE DANS BIBLIOTHEQUE($vgImage;$alResNum{$vlElem};$vsNom)
       Fin de si
    Fin de boucle
    FERMER FENETRE
```

Référence

LISTE TYPES RESSOURCE.

LISTE DE CHAINES VERS TABLEAU

version 6.0

LISTE DE CHAINES VERS TABLEAU (resNum; chaînes{; resFichier})

Paramètre	Type		Description
resNum	Numérique	→	Numéro de ressource
chaînes	Tableau alpha	→	Tableau texte ou alpha devant recevoir les chaînes
		←	Chaînes de la ressource STR#
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou tous les fichiers de ressources ouverts si ce paramètre est omis

Description

La commande LISTE DE CHAINES VERS TABLEAU remplit le tableau chaînes avec les chaînes stockées dans la ressource de type liste de chaînes ("STR#") dont vous avez passé le numéro d'ID dans resNum.

Si la ressource n'est pas trouvée, le tableau chaînes reste inchangé et la variable système OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est recherchée dans ce fichier uniquement. Si vous ne passez pas le paramètre resFichier, c'est la première occurrence de la ressource rencontrée dans la chaîne des fichiers de ressources qui sera retournée.

Si vous ne pré-déclarez pas le tableau chaînes avant d'appeler LISTE DE CHAINES VERS TABLEAU, la commande crée un tableau de type Texte. Si vous pré-déclarez le tableau, vous pouvez lui assigner le type Alpha ou Texte.

Note : Chaque chaîne d'une ressource liste de chaînes peut contenir jusqu'à 255 caractères.

Conseil : Lorsque vous utilisez des ressources listes de chaînes, limitez-vous à des ressources de 32 Ko maximum et quelques centaines de chaînes par ressource.

Exemple

Reportez-vous à l'exemple de la commande TABLEAU VERS LISTE DE CHAINES.

Référence

Lire chaine dans liste, Lire ressource chaine, Lire ressource texte, TABLEAU VERS LISTE DE CHAINES.

Variables et ensembles système

La variable système OK prend la valeur 1 si la ressource est trouvée, sinon elle prend la valeur 0 (zéro).

TABLEAU VERS LISTE DE CHAINES (chaînes; resNum{; resFichier})

Paramètre	Type		Description
chaînes	Tableau alpha	→	Tableau alpha ou texte (nouveau contenu de la ressource STR#)
resNum	Numérique	→	Numéro de ressource
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou fichier de ressources courant si ce paramètre est omis

Description

La commande TABLEAU VERS LISTE DE CHAINES crée ou réécrit la ressource liste de chaînes ("STR#") dont vous avez passé le numéro dans resNum. Le contenu de la ressource est créé à partir des chaînes de caractères que vous avez passées dans le tableau chaînes. Le tableau peut être de type Alpha ou Texte.

Si la ressource ne peut être ajoutée, la variable système OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est ajoutée dans ce fichier. Si vous ne passez pas le paramètre resFichier, la ressource est ajoutée au premier des fichiers de la chaîne des fichiers de ressources (c'est-à-dire le dernier fichier de ressources ouvert).

Note : Chaque chaîne d'une ressource liste de chaînes peut contenir jusqu'à 255 caractères.

Conseil : Lorsque vous utilisez des ressources listes de chaînes, limitez-vous à des ressources de 32 Ko maximum et quelques centaines de chaînes par ressource.

Exemple

Votre base s'appuie sur un même ensemble de polices. Dans la Méthode base Sur fermeture, vous pouvez écrire :

```
      ` Méthode base Sur fermeture
   Si (<>vbPolicesOK)
      LISTE DES POLICES($atFont)
      $vhResFile:=Ouvrir fichier ressources("EnsemblePolices")
      Si (OK=1)
⇒        TABLEAU VERS LISTE DE CHAINES($atFont;15000;$vhResFile)
         FERMER FICHIER RESSOURCES($vhResFile)
      Fin de si
   Fin de si
```

Dans la Méthode base Sur ouverture, vous pouvez écrire :

```
   ` Méthode base Sur ouverture
<>vbPolicesOK:=Faux
LISTE DES POLICES($atNewFont)
Si (Tester chemin acces("EnsemblePolices")#Est un document)
   $vhResFile:=Creer fichier ressources("EnsemblePolices")
Sinon
   $vhResFile:=Ouvrir fichier ressources("EnsemblePolices")
Fin de si
Si (OK=1)
   LISTE DE CHAINES VERS TABLEAU(15000;$atOldFont;$vhResFile)
   Si (OK=1)
      <>vbFontsAreOK:=Vrai
      Boucle($vlElem;1;Taille tableau($atNewFont))
         Si ($atNewFont{$vlElem}#$atOldFont{$vlElem}))
            $vlElem:=MAXLONG
            <>vbPolicesOK:=Faux
         Fin de si
      Fin de boucle
   Sinon
      <>vbPolicesOK:=Vrai
   Fin de si
   FERMER FICHIER RESSOURCES($vhResFile)
Fin de si
Si(Non(<>vbPolicesOK))
   CONFIRMER("Vous n'utilisez pas le même ensemble de polices, OK?")
   Si(OK=1)
      <>vbPolicesOK:=Vrai
   Sinon
      QUITTER 4D
   Fin de si
Fin de si
```

Référence

ECRIRE RESSOURCE CHAINE, ECRIRE RESSOURCE TEXTE, LISTE DE CHAINES VERS TABLEAU.

Variables et ensembles système

La variable système OK prend la valeur 1 si la ressource a été écrite, sinon elle prend la valeur 0 (zéro).

Lire chaine dans liste

Ressources

version 6.0

Lire chaine dans liste (resNum; strNum{; resFichier}) → Alpha

Paramètre	Type		Description
resNum	Numérique	→	Numéro de ressource
strNum	Numérique	→	Numéro de chaîne
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou tous les fichiers de ressources ouverts si ce paramètre est omis
Résultat	Alpha	←	Valeur de la chaîne indexée

Description

La commande Lire chaine dans liste retourne une des chaînes stockées dans la ressource liste de chaînes ("STR#") dont vous avez passé le numéro d'ID dans resNum.

Vous passez le numéro de la chaîne dans strNum. Les chaînes d'une ressource liste de chaînes sont numérotées de 1 à N. Pour récupérer toutes les chaînes (et donc leur nombre) d'une ressource liste de chaînes, utilisez la commande LISTE DE CHAINES VERS TABLEAU.

Si la ressource n'est pas trouvée, ou si la chaîne n'est pas trouvée à l'intérieur de la ressource, une chaîne vide est retournée et la variable système OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est recherchée dans ce fichier uniquement. Si vous ne passez pas le paramètre resFichier, c'est la première occurrence de la ressource rencontrée dans la chaîne des fichiers de ressources qui sera retournée.

Note : Chaque chaîne d'une ressource liste de chaînes peut contenir jusqu'à 255 caractères.

Exemple

Reportez-vous à l'exemple de la commande Mois de.

Référence

Lire ressource chaine, Lire ressource texte, LISTE DE CHAINES VERS TABLEAU.

Variables et ensembles système

OK prend la valeur 1 si la ressource est trouvée, sinon elle prend la valeur 0 (zéro).

Lire ressource chaine

Lire ressource chaine (resNum{; resFichier}) → Alpha

Paramètre	Type		Description
resNum	Numérique	→	Numéro de ressource
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou tous les fichiers de ressources ouverts si ce paramètre est omis
Résultat	Alpha	←	Contenu de la ressource STR

Description

La commande Lire ressource chaine retourne la chaîne stockée dans la ressource chaîne ("STR ") dont vous avez passé le numéro d'ID dans resNum.

Si la ressource n'est pas trouvée, une chaîne vide est retournée et la variable système OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est recherchée dans ce fichier uniquement. Si vous ne passez pas le paramètre resFichier, c'est la première occurrence de la ressource rencontrée dans la chaîne des fichiers de ressources qui sera retournée.

Note : Une ressource chaîne peut contenir jusqu'à 255 caractères.

Exemple

L'exemple suivant affiche le contenu de la ressource chaîne d'ID=20911 qui doit se trouver dans au moins un des fichiers de ressources ouverts :

```
⇒ ALERTE (Lire ressource chaine(20911))
```

Référence

ECRIRE RESSOURCE CHAINE, Lire chaine dans liste, Lire ressource texte, LISTE DE CHAINES VERS TABLEAU.

Variables et ensembles système

La variable système OK prend la valeur 1 si la ressource est trouvée, sinon elle prend la valeur 0 (zéro).

ECRIRE RESSOURCE CHAINE

Ressources

version 6.0

ECRIRE RESSOURCE CHAINE (resNum; resDonnées{; resFichier})

Paramètre	Type		Description
resNum	Numérique	→	Numéro de ressource
resDonnées	Alpha	→	Nouveau contenu de la ressource STR
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou fichier de ressources courant si ce paramètre est omis

Description

La commande ECRIRE RESSOURCE CHAINE crée ou réécrit la ressource chaîne ("STR ") dont vous avez passé le numéro dans resNum avec la chaîne de caractères que vous avez passée dans resDonnées.

Si la ressource ne peut être ajoutée, la variable système OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est ajoutée dans ce fichier. Si vous ne passez pas le paramètre resFichier, la ressource est ajoutée au premier des fichiers de la chaîne des fichiers de ressources (c'est-à-dire le dernier fichier de ressources ouvert).

Note : Une ressource chaîne peut contenir jusqu'à 255 caractères.

Référence

ECRIRE RESSOURCE TEXTE, Lire ressource chaine.

Variables et ensembles système

La variable système OK prend la valeur 1 si la ressource a été écrite, sinon elle prend la valeur 0 (zéro).

Lire ressource texte

version 6.0

Lire ressource texte (resNum{; resFichier}) → Texte

Paramètre	Type		Description
resNum	Numérique	→	Numéro de ressource
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou tous les fichiers de ressources ouverts si ce paramètre est omis
Résultat	Texte	←	Contenu de la ressource TEXT

Description

La commande Lire ressource texte retourne le texte stocké dans la ressource texte ("TEXT") dont vous avez passé le numéro d'ID dans resNum.

Si la ressource n'est pas trouvée, un texte vide est retourné et la variable système OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est recherchée dans ce fichier uniquement. Si vous ne passez pas le paramètre resFichier, c'est la première occurrence de la ressource rencontrée dans la chaîne des fichiers de ressources qui sera retournée.

Note : Une ressource texte peut contenir jusqu'à 32 000 caractères.

Exemple

L'exemple suivant affiche le contenu de la ressource texte d'ID=20800 qui doit se trouver dans au moins un des fichiers de ressources ouverts :

```
⇒ ALERTE (Lire ressource texte(20800))
```

Référence

ECRIRE RESSOURCE TEXTE, Lire chaine dans liste, Lire ressource chaine, LISTE DE CHAINES VERS TABLEAU.

Variables et ensembles système

OK prend la valeur 1 si la ressource est trouvée, sinon elle prend la valeur 0 (zéro).

ECRIRE RESSOURCE TEXTE

ECRIRE RESSOURCE TEXTE (resNum; resDonnées{; resFichier})

Paramètre	Type		Description
resNum	Numérique	→	Numéro de ressource
resDonnées	Alpha	→	Nouveau contenu de la ressource TEXT
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou fichier de ressources courant si ce paramètre est omis

Description

La commande ECRIRE RESSOURCE TEXTE crée ou réécrit la ressource texte ("TEXT") dont vous avez passé le numéro dans resNum avec le texte ou la chaîne de caractères que vous avez passée dans resDonnées.

Si la ressource ne peut être ajoutée, la variable système OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est ajoutée dans ce fichier. Si vous ne passez pas le paramètre resFichier, la ressource est ajoutée au premier des fichiers de la chaîne des fichiers de ressources (c'est-à-dire le dernier fichier de ressources ouvert).

Note : Une ressource texte peut contenir jusqu'à 32 000 caractères.

Référence

ECRIRE RESSOURCE CHAINE, Lire ressource texte.

Variables et ensembles système

La variable système OK prend la valeur 1 si la ressource a été écrite, sinon elle prend la valeur 0 (zéro).

LIRE RESSOURCE IMAGE (resNum; resDonnées{; resFichier})

Paramètre	Type		Description
resNum	Numérique	→	Numéro de ressource
resDonnées	Champ | Variable	→	Champ ou variable image devant recevoir l'image
		←	Contenu de la ressource PICT
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou tous les fichiers de ressources ouverts si ce paramètre est omis

Description

La commande LIRE RESSOURCE IMAGE retourne dans le champ ou la variable image désigné(e) par resDonnées l'image stockée dans la ressource image ("PICT") dont vous passé le numéro dans resNum.

Si la ressource n'est pas trouvée, resDonnées n'est pas modifié et la variable OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est recherchée dans ce fichier uniquement. Si vous ne passez pas le paramètre resFichier, c'est la première occurrence de la ressource rencontrée dans la chaîne des fichiers de ressources qui sera retournée.

Note : La taille d'une ressource image peut atteindre plusieurs méga-octets, voire davantage.

Exemple

Reportez-vous à l'exemple de la commande LISTE RESSOURCES.

Référence

APPELER SUR ERREUR, ECRIRE RESSOURCE IMAGE, LIRE RESSOURCE ICONE.

Variables et ensembles système

La variable système OK prend la valeur 1 si la ressource est trouvée, sinon elle prend la valeur 0 (zéro).

Gestion des erreurs

S'il n'y a pas assez de mémoire disponible pour charger l'image, une erreur est générée. Vous pouvez intercepter cette erreur à l'aide d'une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

ECRIRE RESSOURCE IMAGE (resNum; resDonnées{; resFichier})

Paramètre	Type		Description
resNum	Numérique	→	Numéro de ressource
resDonnées	Image	→	Nouveau contenu de la ressource PICT
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou fichier de ressources courant si ce paramètre est omis

Description

La commande ECRIRE RESSOURCE IMAGE crée ou réécrit la ressource image ("PICT") dont vous avez passé le numéro dans resNum avec l'image que vous avez passée dans resDonnées.

Si la ressource ne peut être créée, la variable système OK prend la valeur 0 (zéro).

Note : Si vous passez dans resDonnées un champ ou une variable image vide (c'est-à-dire si sa taille est nulle), la commande ne fait rien et la variable OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est ajoutée dans ce fichier. Si vous ne passez pas le paramètre resFichier, la ressource est ajoutée au premier des fichiers de la chaîne des fichiers de ressources (c'est-à-dire le dernier fichier de ressources ouvert).

Note : La taille d'une ressource image peut atteindre plusieurs méga-octets, voire davantage.

Référence

LIRE RESSOURCE IMAGE.

Variables et ensembles système

La variable système OK prend la valeur 1 si la ressource a été écrite, sinon elle prend la valeur 0 (zéro).

LIRE RESSOURCE ICONE (resNum; imageDest{; resFichier})

Paramètre	Type		Description
resNum	Numérique	→	Numéro de ressource icône
imageDest	Image	←	Image résultante
resFichier	Numérique	→	Numéro de référence de fichier de ressources ou tous les fichiers de ressources ouverts si ce paramètre est omis

Description

La commande LIRE RESSOURCE ICONE retourne dans le champ ou la variable image imageDest l'icône stockée dans la ressource icône couleur ("cicn") dont vous avez passé le numéro d'ID dans resNum.

Si la ressource n'est pas trouvée, le paramètre imageDest reste inchangé et la variable système OK prend la valeur 0 (zéro).

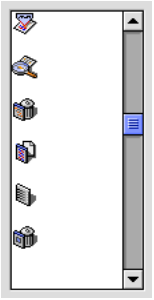
Si vous passez un numéro de référence de fichier de ressources valide dans resFichier, la ressource est recherchée dans ce fichier uniquement. Si vous ne passez pas le paramètre resFichier, c'est la première occurrence de la ressource rencontrée dans la chaîne des fichiers de ressources qui sera retournée.

Exemple

L'exemple suivant charge dans un tableau image les icônes couleur situées dans l'application 4D en cours d'utilisation :

```
Si (Sous Windows)
   $vh4DResFile:=Ouvrir fichier ressources(Remplacer chaine(Fichier
                                                  application;".EXE";".RSR"))
Sinon
   $vh4DResFile:=Ouvrir fichier ressources(Fichier application)
Fin de si
LISTE RESSOURCES("cicn";$alResID;$asResNom;$vh4DResFile)
$vlNbIcons:=Taille tableau($alResID)
TABLEAU IMAGE(ag4DIcon;$vlNbIcons)
Boucle (1;$vlNbIcons)
⇒     LIRE RESSOURCE ICONE($alResID{$vlElem};ag4DIcon{$vlElem};$vh4DResFile)
Fin de boucle
```

Une fois ce code exécuté, le tableau aura l'aspect suivant lorsqu'il sera affiché dans un formulaire :



Référence

LIRE RESSOURCE IMAGE.

Variables et ensembles système

La variable système OK prend la valeur 1 si la ressource est trouvée, sinon elle prend la valeur 0 (zéro).

LIRE RESSOURCE (typeRes; numRes; donnéesRes{; fichierRes})

Paramètre	Type		Description
typeRes	Alpha	→	Type de ressource (4 caractères)
numRes	Numérique	→	Numéro de ressource
donnéesRes	BLOB	→	Champ ou variable BLOB devant recevoir les données
		←	Contenu de la ressource
fichierRes	DocRef	→	Numéro de référence de fichier de ressources ou tous les fichiers de ressources ouverts si ce paramètre est omis

Description

La commande LIRE RESSOURCE retourne dans le champ ou la variable BLOB donnéesRes le contenu de la ressource dont le type et le numéro sont passés dans typeRes et numRes.

Important : Vous devez passer une chaîne de 4 caractères dans typeRes.

Si la ressource n'est pas trouvée, le paramètre donnéesRes est laissé inchangé et la variable OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans fichierRes, la ressource sera recherchée dans ce fichier seulement. Si ne passez pas le paramètre fichierRes, la première occurrence de la ressource trouvée en remontant la chaîne des fichiers de ressources sera retournée.

Note : La taille d'une ressource peut atteindre plusieurs méga-octets.

Indépendance de plate-forme : Rappelez-vous que vous travaillez avec des ressources issues de MacOS. Quelle que soit la plate-forme utilisée, les valeurs internes des ressources comme des entiers longs sont stockées avec l'ordre d'octets ("byte ordering") MacOS. Sous Windows, pour les données des ressources standard (telles que les ressources listes de chaînes et les ressources images) l'ordre des octets est automatiquement inversé ("byte swapping") si nécessaire. D'un autre côté, si vous créez et utilisez vos propres structures internes de données, c'est à vous d'effectuer l'inversion d'octets des données lorsque vous les extrayez d'un BLOB (par exemple en passant Ordre octets Macintosh à une commande telle que BLOB vers entier long).

Exemple

Reportez-vous à l'exemple de la commande ECRIRE RESSOURCE.

Référence

Commandes du thème BLOB, ECRIRE RESSOURCE, Ressources.

Variables et ensembles système

Si la ressource est trouvée, la variable système OK prend la valeur 1. Sinon, elle prend la valeur 0 (zéro).

Gestion des erreurs

S'il n'y a pas assez de mémoire disponible pour charger l'image, une erreur est générée. Vous pouvez intercepter cette erreur à l'aide d'une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

ECRIRE RESSOURCE (typeRes; numRes; donnéesRes{; fichierRes})

Paramètre	Type		Description
typeRes	Alpha	→	Type de ressource (4 caractères)
numRes	Numérique	→	Numéro de ressource
donnéesRes	BLOB	→	Nouveau contenu de la ressource
fichierRes	DocRef	→	Numéro de référence de fichier de ressources ou Fichier de ressources courant si omis

Description

La commande ECRIRE RESSOURCE crée ou réécrit la ressource dont vous avez passé le type et le numéro dans typeRes et numRes avec les données passées dans le BLOB donnéesRes.

Important : Vous devez passer une chaîne de 4 caractères dans typeRes.

Si la ressource ne peut pas être écrite, la variable système OK prend la valeur 0 (zéro).

Si vous passez un numéro de référence de fichier de ressources valide dans fichierRes, la ressource est ajoutée dans ce fichier. Si vous ne passez pas le paramètre fichierRes, la ressource est ajoutée au premier des fichiers de la chaîne des fichiers de ressources (c'est-à-dire le dernier fichier de ressources ouvert).

Note : La taille d'une ressource peut atteindre plusieurs mega-octets.

Indépendance de plate-forme : Rappelez-vous que vous travaillez avec des ressources issues de MacOS. Quelle que soit la plate-forme utilisée, les valeurs internes des ressources comme des entiers longs sont stockées avec l'ordre d'octets ("byte ordering") MacOS. Sous Windows, pour les données des ressources standard (telles que les ressources listes de chaînes et les ressources images) l'ordre des octets est automatiquement inversé ("byte swapping") si nécessaire. D'un autre côté, si vous créez et utilisez vos propres structures internes de données, c'est à vous d'effectuer l'inversion d'octets des données lorsque vous les écrivez dans un BLOB (par exemple en passant Ordre octets Macintosh à une commande telle que ENTIER LONG VERS BLOB).

Exemple

Pendant une session 4D, vous conservez des préférences utilisateur dans des variables interprocess. Pour sauvegarder ces préférences d'une session sur l'autre, vous pouvez :

• utiliser les commandes ECRIRE VARIABLES et LIRE VARIABLES pour stocker et récupérer les variables dans des documents de variables sur disque.

• utiliser les commandes VARIABLE VERS BLOB, BLOB VERS DOCUMENT, DOCUMENT VERS BLOB et BLOB VERS VARIABLE pour stocker et récupérer les variables dans des documents BLOB sur disque.

• utiliser les commandes VARIABLE VERS BLOB, ECRIRE RESSOURCE, LIRE RESSOURCE et BLOB VERS VARIABLE pour stocker et récupérer les variables dans des fichiers de ressources sur disque.

L'exemple suivant utilise la troisième possibilité :

Dans la Méthode base Sur fermeture, vous écrivez :

```
      ` Méthode base Sur fermeture
   Si (Tester chemin acces("DB_Prefs")#Est un document)
      $vhResFile:=Creer fichier ressources("DB_Prefs")
   Sinon
      $vhResFile:=Ouvrir fichier ressources("DB_Prefs")
   Fin de si
   Si (OK=1)
      VARIABLE VERS BLOB(<>vbAutoRepeat;$vxPrefData)
      VARIABLE VERS BLOB(<>vlCurTable;$vxPrefData;*)
      VARIABLE VERS BLOB(<>asDfltOption;$vxPrefData;*)
         ` et ainsi de suite...
⇒     ECRIRE RESSOURCE("PREF";26500;$vxPrefData;$vhResFile)
      FERMER FICHIER RESSOURCES($vhResFile)
   Fin de si
```

Dans la Méthode base Sur ouverture, vous écrivez :

```
      ` Méthode base Sur ouverture
   C_BOOLEEN(<>vbAutoRepeat)
   C_ENTIER LONG(<>vlCurTable)
   $vbDone:=Faux
   $vhResFile:=Ouvrir fichier ressources("DB_Prefs")
   Si (OK=1)
⇒     LIRE RESSOURCE("PREF";26500;$vxPrefData;$vhResFile)
      Si (OK=1)
         $vlOffset:=0
         BLOB VERS VARIABLE($vxPrefData;<>vbAutoRepeat;$vlOffset)
         BLOB VERS VARIABLE($vxPrefData;<>vlCurTable;$vlOffset)
         BLOB VERS VARIABLE($vxPrefData;<>asDfltOption;$vlOffset)
            ` et ainsi de suite...
         $vbDone:=Faux
      Fin de si
      FERMER FICHIER RESSOURCES($vhResFile)
   Fin de si
   Si(Non($vbDone))
      <>vbAutoRepeat:=Faux
      <>vlCurTable:=0
      TABLEAU ALPHA(127;<>asDfltOption;0)
   Fin de si
```

Référence

Commandes du thème BLOB, LIRE RESSOURCE.

Variables et ensembles système

Si la ressource est écrite, la variable système OK prend la valeur 1. Sinon, elle prend la valeur 0 (zéro).

Lire nom ressource

Ressources

version 6.0

Lire nom ressource (typeRes; resID{; fichierRes}) → Alpha

Paramètre	Type		Description
typeRes	Alpha	→	Type de ressource (4 caractères)
resID	Numérique	→	Numéro de référence de ressource (ID)
fichierRes	RefDoc	→	Numéro de référence du fichier de ressources ou Tous les fichiers de ressources ouverts si ce paramètre est omis
Résultat	Alpha	←	Nom de la ressource

Description

Lire nom ressource retourne le nom de la ressource dont le type est passé dans typeRes et le numéro de référence (ID) dans resID.

Si vous ne passez pas le paramètre fichierRes, la ressource est recherchée dans tous les fichiers de ressources ouverts. Si vous passez un numéro de référence de fichier de ressource dans le paramètre fichierRes, la ressource n'est recherchée que dans ce fichier.

Si la ressource n'existe pas, Lire nom ressource retourne une chaîne vide et fixe la variable OK à 0 (zéro).

Exemple

La méthode projet suivante copie une ressource ainsi que son nom et ses attributs d'un fichier de ressources vers un autre :

```
` Méthode projet COPIER RESSOURCE
` COPIER RESSOURCE ( Alpha ; Entier long ; Heure ; Heure )
` COPIER RESSOURCE ( typeRes ; IDRes ; fichierResSource ; fichierResDest )

C_ALPHA (4;$1)
C_ENTIER LONG ($2)
C_HEURE ($3;$4)
C_BLOB ($vxResData)
LIRE RESSOURCE ($1;$2;$vxData;$3)
```

```
   Si (OK=1)
      ECRIRE RESSOURCE ($1;$2;$vxData;$4)
      Si (OK=1)
⇒        ECRIRE NOM RESSOURCE ($1;$2; Lire nom ressource ($1;$2;$3);$4)
         ECRIRE PROPRIETES RESSOURCE ($1;$2; Lire proprietes
                                                    ressource ($1;$2;$3);$4)
      Fin de si
   Fin de si
```

Lorsque cette méthode projet est présente dans votre application, vous pouvez écrire :

```
      ` Copier la ressource 'DATA' ID = 15000 de fichier A au fichier B
   COPIER RESSOURCE ("DATA";15000;$vhFichResA;$vhFichResB)
```

Référence

ECRIRE PROPRIETES RESSOURCE.

Variables et ensembles système

La variable OK prend la valeur 0 si la ressource n'existe pas, sinon elle prend la valeur 1.

ECRIRE NOM RESSOURCE

version 6.0

ECRIRE NOM RESSOURCE (typeRes; resID; nomRes{; fichierRes})

Paramètre	Type		Description
typeRes	Alpha	→	Type de ressource (4 caractères)
resID	Numérique	→	Numéro de référence de ressource (ID)
nomRes	Alpha	→	Nouveau nom de la ressource
fichierRes	RefDoc	→	Numéro de référence du fichier de ressources ou Tous les fichiers de ressources ouverts si ce paramètre est omis

Description

ECRIRE NOM RESSOURCE modifie le nom de la ressource dont le type est passé dans typeRes et le numéro de référence dans resID.

Si vous ne passez pas le paramètre fichierRes, la ressource est recherchée dans les fichiers de ressources ouverts. Si vous passez un numéro de référence de fichier de ressource dans le paramètre fichierRes, la ressource n'est recherchée que dans ce fichier.

Si la ressource n'existe pas, ECRIRE NOM RESSOURCE ne fait rien et la variable OK prend la valeur 0 (zéro).

ATTENTION : Il est déconseillé de modifier les noms des ressources appartenant à 4D ou aux fichiers système. Cette opération peut provoquer des erreurs système.

Note : Les noms des ressources peuvent comprendre jusqu'à 255 caractères. Ils n'établissent pas de distinction entre les majuscules et les minuscules, mais les caractères diacritiques (é, è, etc.) sont respectés.

Exemple

Référez-vous à l'exemple de la commande Lire nom ressource.

Référence

ECRIRE PROPRIETES RESSOURCE.

Variables et ensembles système

La variable système OK prend la valeur 0 si la ressource n'existe pas, sinon elle prend la valeur 1.

Lire proprietes ressource

Ressources

version 6.0

Lire proprietes ressource (typeRes; resID{; fichierRes}) → Numérique

Paramètre	Type		Description
typeRes	Alpha	→	Type de ressource (4 caractères)
resID	Numérique	→	Numéro de référence de ressource (ID)
fichierRes	RefDoc	→	Numéro de référence du fichier de ressources ou Tous les fichiers de ressources ouverts si ce paramètre est omis
Résultat	Numérique	←	Attributs de la ressource

Description

Lire proprietes ressource retourne les attributs de la ressource dont le type est passé dans le paramètre typeRes et le numéro de référence dans resID.

Si vous ne passez pas le paramètre fichierRes, la ressource est recherchée dans les fichiers de ressources ouverts. Si vous passez un numéro de référence de fichier de ressource dans le paramètre fichierRes, la ressource n'est recherchée que dans ce fichier.

Si la ressource n'existe pas, Lire proprietes ressource retourne 0 (zéro) et la variable OK prend également la valeur 0 (zéro).

La valeur numérique retournée par Lire proprietes ressource doit être considérée comme une valeur binaire dont chaque bit a une signification particulière. Pour une description des attributs des ressources et leurs effets, référez-vous à la commande ECRIRE PROPRIETES RESSOURCE.

Exemple

Référez-vous à l'exemple de la commande Lire nom ressource.

Référence

ECRIRE NOM RESSOURCE.

Variables et ensembles système

La variable OK prend la valeur 0 si la ressource n'existe pas, sinon elle prend la valeur 1.

ECRIRE PROPRIETES RESSOURCE

ECRIRE PROPRIETES RESSOURCE (typeRes; resID; attrRes{; fichierRes})

Paramètre	Type		Description
typeRes	Alpha	→	Type de ressource (4 caractères)
resID	Numérique	→	Numéro de référence de ressource (ID)
attrRes	Numérique	→	Nouveaux attributs de la ressource
fichierRes	RefDoc	→	Numéro de référence du fichier de ressources ou Tous les fichiers de ressources ouverts si ce paramètre est omis

Description

ECRIRE PROPRIETES RESSOURCE modifie les attributs de la ressource dont le type est passé dans le paramètre typeRes et le numéro de référence dans resID.

Si vous ne passez pas le paramètre fichierRes, la ressource est recherchée dans les fichiers de ressources ouverts. Si vous passez un numéro de référence de fichier de ressource dans le paramètre fichierRes, la ressource n'est recherchée que dans ce fichier.

Si la ressource n'existe pas, ECRIRE PROPRIETES RESSOURCE ne fait rien et la variable OK prend la valeur 0 (zéro).

La valeur numérique passée dans attrRes doit être considérée comme une valeur binaire dont chaque bit a une signification particulière. Les constantes ci-dessous sont fournies par 4e Dimension :

Constante	Type	Valeur
Masque ressource heap système	Entier long	64
Bit ressource heap système	Entier long	6
Masque ressource purgeable	Entier long	32
Bit ressource purgeable	Entier long	5
Masque ressource verrouillée	Entier long	16
Bit ressource verrouillée	Entier long	4
Masque ressource protégée	Entier long	8
Bit ressource protégée	Entier long	3
Masque ressource préchargée	Entier long	4
Bit ressource préchargée	Entier long	2
Masque ressource modifiée	Entier long	2
Bit ressource modifiée	Entier long	1

A l'aide de ces constantes, vous pouvez définir la valeur d'attributs de ressources que vous voulez. Référez-vous aux exemples proposés à la fin de cette section.

Les attributs de ressources et leurs effets sont décrits ci-dessous :

Heap système
Si cet attribut est utilisé, la ressource sera chargée dans la mémoire système au lieu de la mémoire de 4D. Vous ne devez pas utiliser cet attribut, sauf si vous êtes absolument sûr de savoir très exactement ce que vous faites.

Purgeable
Lorsque cet attribut est utilisé, la ressource chargée peut à tout moment être purgée (c'est-à-dire supprimée) de la mémoire si de la place mémoire est nécessaire pour l'allocation d'autres données. Comme de toute manière vous chargez des ressources dans des BLOBs 4D, nous vous conseillons de rendre purgeables toutes vos propres ressources afin d'optimiser l'utilisation de la mémoire. Notez cependant que si une ressource est fréquemment appelée lors d'une session de travail, il peut être intéressant de la rendre non purgeable afin de réduire les accès disque dûs au rechargement de la ressource purgée.

Verrouillée
Si cet attribut est utilisé, la ressource ne peut pas être déplacée lorsqu'elle est chargée en mémoire. Une ressource verrouillée ne peut pas être purgée même si elle est purgeable. Le verrouillage d'une ressource a l'effet indésirable de fragmenter la mémoire. Vous ne devez pas utiliser cet attribut, sauf si vous êtes absolument sûr de savoir très exactement ce que vous faites.

Protégée
Si cet attribut est utilisé, vous ne pouvez plus modifier le nom, le numéro de référence (ID) ou le contenu de la ressource. Il n'est également plus possible de supprimer cette ressource. Vous pouvez toutefois appeler la commande ECRIRE PROPRIETES RESSOURCE pour supprimer cet attribut, puis modifier ou supprimer la ressource. La plupart du temps, vous n'aurez pas à utiliser cet attribut.

Note : Cet attribut est sans effet sous Windows.

Préchargée
Si cet attribut est utilisé, la ressource est automatiquement chargée en mémoire et le fichier dans lequel elle se trouve est ouvert. Cet attribut est utile pour optimiser le chargement des ressources lors de l'ouverture d'un fichier de ressources. Cependant, la plupart du temps, vous n'aurez pas besoin de cet attribut.

Modifiée
Si cet attribut est utilisé, la ressource reçoit une marque signifiant “doit être sauvegardée sur disque” lorsque le fichier de ressources dans lequel elle se trouve est refermé. Comme la commande de 4D ECRIRE RESSOURCE gère en interne l'écriture et la ré-écriture des ressources, vous ne devez pas utiliser cet attribut, sauf si vous êtes absolument sûr de savoir très exactement ce que vous faites.

En résumé, à moins d'être certain de savoir ce que vous faites, vous n'utiliserez généralement que l'attribut purgeable, ainsi que, plus rarement, les attributs préchargée et protégée.

ATTENTION : Il est déconseillé de modifier les noms des ressources appartenant à 4D ou aux fichiers système. Cette opération peut provoquer des erreurs systèmes.

Exemples

(1) Référez-vous à l'exemple de la fonction Lire nom ressource.

(2) L'exemple suivant rend la ressource 'STR#' ID=17000 purgeable et laisse les autres attributs inchangés :

```
$vlAttrRes:=Lire proprietes ressource ('STR#';17000;$vhFichierRes)
⇒ ECRIRE PROPRIETES RESSOURCE('STR#';17000;$vlAttrRes ?+ Bit ressource purgeable;
                                                        $vhMonFichRes)
```

(3) L'exemple suivant rend la ressource 'STR#' ID=17000 préchargée et non-purgeable :

```
⇒ ECRIRE PROPRIETES RESSOURCE('STR#';17000;Masque ressource préchargée;
                                                        $vhFichierRes)
```

(4) L'exemple suivant rend la ressource 'STR#' ID=17000 préchargée, mais purgeable:

```
⇒ ECRIRE PROPRIETES RESSOURCE('STR#';17000;Masque ressource préchargée+
                                Masque ressource purgeable;$vhFichierRes)
```

Référence

ECRIRE NOM RESSOURCE.

Variables et ensembles système

La variable système OK prend la valeur 0 si la ressource n'existe pas, sinon elle prend la valeur 1.

SUPPRIMER RESSOURCE

Ressources

version 6.0

SUPPRIMER RESSOURCE (resType; resNum{; resFichier})

Paramètre	Type		Description
resType	Alpha	→	Type de ressource (4 caractères)
resNum	Numérique	→	Numéro de ressource
resFichier	DocRef	→	Numéro de référence de fichier de ressources ou Fichier de ressources courant si ce paramètre est omis

Description

La commande SUPPRIMER RESSOURCE supprime la ressource dont vous passez le type dans le paramètre resType et le numéro dans resNum.

Si vous passez un numéro de référence de fichier de ressources dans le paramètre resFichier, 4D recherche la ressource dans ce fichier uniquement. Si vous ne passez pas le paramètre resFichier, 4D recherche la ressource dans tous les fichiers de ressources ouverts.

Si la ressource n'existe pas, SUPPRIMER RESSOURCE ne fait rien et la variable OK prend la valeur 0 (zéro). Si la ressource est correctement identifiée et effacée, la variable système OK prend la valeur 1.

ATTENTION : Ne supprimez pas de ressources appartenant à 4D ou à un fichier du système. Cette opération peut provoquer l'apparition d'erreurs système.

Exemples

(1) L'exemple suivant supprime la ressource "STR#" d'ID=20000:

```
` Notez que cet exemple supprimera la première ressource "STR#" ID=20000
` rencontrée parmi tous les fichiers de ressources actuellement ouverts :
⇒ SUPPRIMER RESSOURCE ("STR#";20000)
```

(2) L'exemple suivant supprime la ressource "STR#" d'ID=20000 si celle-ci est présente dans un fichier particulier :

```
` Notez que cet exemple supprimera la ressource "STR#" d'ID=20000
` seulement si elle est présente dans le fichier de ressources désigné par $vhResFile :
⇒ SUPPRIMER RESSOURCE ("STR#";20000;$vhResFile)
` Notez également que si une ressource identique existe dans un fichier de ressources
` ouvert autre que le fichier spécifié par vhResFile, elle restera inchangée
```

(3) La méthode projet SUPPRIMER RESSOURCES DE TYPE supprime du fichier de ressources spécifié par le premier paramètre toutes les ressources du type spécifié par le second paramètre :

```
      ` Méthode projet SUPPRIMER RESSOURCES DE TYPE
      ` SUPPRIMER RESSOURCES DE TYPE ( Heure ; Alpha )
      ` SUPPRIMER RESSOURCES DE TYPE ( resFichier ; resType )
   C_HEURE($1)
   C_ALPHA(4;$2)
   LISTE RESSOURCES($2;$aiResID;$asResNom;$1)
   Si(OK=1)
      Boucle($vlElem;1;Taille tableau($aiResID))
⇒        SUPPRIMER RESSOURCE($2;$aiResID{$vlElem};$1)
      Fin de boucle
   Fin de si
```

Une fois que cette méthode projet existe dans votre base, vous pouvez écrire :

```
      ` Supprimer toutes les ressources de type "PREF" du fichier de ressources $vhResFile
   SUPPRIMER RESSOURCES DE TYPE ($vhResFile;"PREF")
```

(4) La méthode projet SUPPRIMER RESSOURCE PAR NOM supprime une ressource (d'un type spécifique) dont vous connaissez le nom :

```
      ` Méthode projet SUPPRIMER RESSOURCE PAR NOM
      ` SUPPRIMER RESSOURCE PAR NOM ( Heure ; Alpha ; Alpha )
      ` SUPPRIMER RESSOURCE PAR NOM ( resFichier ; resType ; resNom )
   C_HEURE($1)
   C_ALPHA(4;$2)
   C_ALPHA(255;$3)
   LISTE RESSOURCES($2;$aiResID;$asResName;$1)
   Si(OK=1)
      $vlElem:=Chercher dans tableau($asResName;$3)
      Si($vlElem>0)
⇒        SUPPRIMER RESSOURCE($2;$aiResID{$vlElem};$1)
      Fin de boucle
   Fin de si
```

Une fois que cette méthode projet existe dans votre base, vous pouvez écrire :

```
      ` Supprimer du fichier de ressources $vhResFile la ressource "PREF" dont le nom
      ` est “Réglages standard” :
   SUPPRIMER RESSOURCE PAR NOM ($vhResFile;"PREF";"Réglages standard")
```

Référence

ECRIRE PROPRIETES RESSOURCE, LISTE RESSOURCES.

Variables et ensembles système

La variable OK prend la valeur 0 si la ressource n'existe pas ; si la ressource a été supprimée, OK prend la valeur 1.

Lire ID ressource composant

Ressources

version 6.7

Lire ID ressource composant (nomComp; typeRes; numResOriginal) → Numérique

Paramètre	Type		Description
nomComp	Alpha (32)	→	Nom du composant référençant la ressource
typeRes	Alpha (4)	→	Type de ressource (4 caractères), STR# ou PICT
numResOriginal	Numérique	→	Numéro original de la ressource, avant installation du composant
Résultat	Numérique	←	Numéro courant de la ressource

Description

La commande Lire ID ressource composant permet aux développeurs de composants de s'assurer que leurs appels de ressources personnalisées de type PICT ou STR# seront correctement effectués, même si les numéros de ces ressources ont été modifiés au moment de l'installation.

En effet, lorsqu'un composant utilisant des ressources est installé par 4D Insider, le programme peut renuméroter automatiquement les nouvelles ressources si des ressources de même numéro existent déjà dans la base.

Note : Pour plus d'infomations sur les composants dans 4e Dimension, reportez-vous à la documentation de 4D Insider.

La commande Lire ID ressource composant permet donc de connaître le numéro courant (réel) de chaque ressource utilisée par un composant, à partir de son type et de son numéro original.

• Passez dans le paramètre nomComp le nom du composant utilisant la ressource.

• Passez dans le paramètre typeRes le type de la ressource (impérativement composé de 4 caractères). La commande Lire ID ressource composant accepte uniquement les ressources de type PICT et STR#.

Note : Les images stockées dans la bibliothèque d'images de 4D ne sont PAS gérées par la commande Lire ID ressource composant. Pour pouvoir utiliser des images de la Bibliothèque d'images dans un composant 4D, vous devez appeler la commande LIRE IMAGE DANS BIBLIOTHEQUE et passer une chaîne (le nom de l'image) en tant que premier paramètre. Pour plus d'informations, reportez-vous à la description de la commande LIRE IMAGE DANS BIBLIOTHEQUE.

• Passez dans numResOriginal le numéro original de la ressource, c'est-à-dire celui défini au moment de la création du composant.

La fonction retourne alors le numéro attribué à la ressource dans l'application courante. Si aucune ressource ne correspond à numResOriginal, Lire ID ressource composant retourne la valeur saisie dans numResOriginal.

Exemple

Le code ci-dessous ne permet pas de garantir que les appels de ressources seront correctement effectués :

```
    `Si les ressources sont renumérotées, cet appel sera incorrect
vNumRes := 15000
LISTE DE CHAINES VERS TABLEAU(vNumRes; tabChaînes; fichierRes)
```

Il est fortement conseillé de préférer le code suivant :

```
    `Cet appel sera correct dans tous les cas
⇒   vNumRes :=Lire ID ressource composant("MonComp";"STR#";15000)
    LISTE DE CHAINES VERS TABLEAU (vNumRes; tabChaînes; fichierRes)
```

Référence

LIRE INFORMATIONS SERIALISATION.

46

Saisie

AJOUTER ENREGISTREMENT ({table}{; }{*})

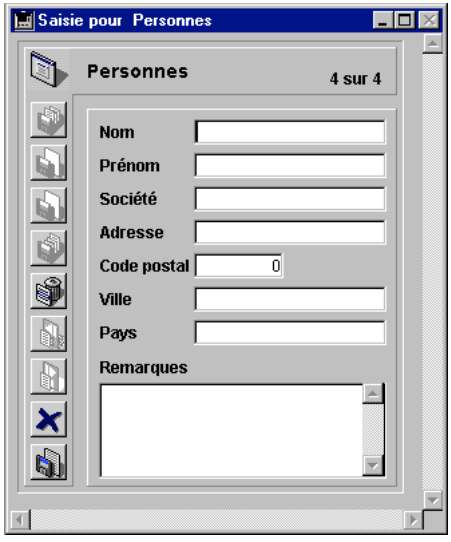
Paramètre	Type		Description
table	Table	→	Table dans laquelle ajouter des données ou Table par défaut si ce paramètre est omis
*		→	Cacher les barres de défilement

Description

La commande AJOUTER ENREGISTREMENT permet à l'utilisateur de créer un nouvel enregistrement dans table ou dans la table par défaut si ce paramètre est omis.

AJOUTER ENREGISTREMENT crée un nouvel enregistrement pour table, en fait l'enregistrement courant pour le process courant et l'affiche dans le formulaire entrée courant. En mode Menus créés, une fois que l'utilisateur a validé le nouvel enregistrement, la sélection courante est réduite à ce seul enregistrement.

L'écran suivant présente un formulaire typiquement utilisé pour la saisie de données :



Le formulaire est affiché dans la fenêtre se trouvant au premier plan du process. Elle comporte des barres de défilement et une case de contrôle de taille. Si vous passez le paramètre optionnel astérisque (*), ces éléments n'apparaîtront pas.

AJOUTER ENREGISTREMENT affiche le formulaire jusqu'à ce que l'utilisateur valide ou annule l'enregistrement. Si l'utilisateur ajoute plusieurs enregistrements, la commande doit être appelée pour chaque nouvel enregistrement.

L'enregistrement est sauvegardé si l'utilisateur clique sur un bouton du type Valider ou appuie sur la touche Entrée, ou encore si la commande VALIDER est exécutée.

L'enregistrement n'est pas sauvegardé si l'utilisateur clique sur un bouton du type Annuler ou appuie sur la touche d'annulation (**Echap** sous Windows, **Esc** sous MacOS), ou encore si la commande NE PAS VALIDER est exécutée.

Après un appel à AJOUTER ENREGISTREMENT, la variable système OK prend la valeur 1 si l'enregistrement est validé et 0 s'il est annulé.

Note : Même lorsqu'il est annulé, l'enregistrement reste en mémoire et peut être sauvegardé avec la commande STOCKER ENREGISTREMENT si celle-ci est exécutée avant que le pointeur d'enregistrement courant ne soit modifié.

Exemples

(1) L'exemple suivant est une boucle souvent utilisée pour créer des enregistrements dans une base :

```
      ` Désigner le formulaire entrée de la table [Clients]
FORMULAIRE ENTREE ([Clients]; "SaisieClients")
Repeter ` Boucle jusqu'à ce que l'utilisateur annule
      ` Ajouter un enregistrement dans la table [Clients]
⇒     AJOUTER ENREGISTREMENT ([Clients];*)
Jusque (OK = 0) ` Jusqu'à ce que l'utilisateur annule
```

(2) L'exemple suivant permet de rechercher un client dans la base. Le déroulement de la méthode dépend du résultat de la recherche. Si aucun client n'a été trouvé, l'utilisateur est autorisé à créer un nouveau client à l'aide de la commande AJOUTER ENREGISTREMENT. Si au moins un client a été trouvé, le premier enregistrement est affiché pour modification, à l'aide de la commande MODIFIER ENREGISTREMENT :

```
LECTURE ECRITURE ([Clients])
FORMULAIRE ENTREE ([Clients]; "Entrée1") `  Désigner le formulaire entrée
   ` On récupère le numéro du client
vlClientNo:=Num(Demander ("Saisissez un numéro de client :")
Si (OK =1)
   CHERCHER ([Clients]; [Clients]ClientNo = vlClientNo) ` Recherche du client
   Si (Enregistrements trouves([Clients]) = 0) ` Si aucun client n'a été trouvé...
⇒     AJOUTER ENREGISTREMENT([Clients]) `Ajout d'un nouveau client
   Sinon
      Si(Non(Enregistrement verrouille([Clients])))
         MODIFIER ENREGISTREMENT([Clients]) `Modifier l'enregistrement
         LIBERER ENREGISTREMENT([Clients])
      Sinon
         ALERTE("Cet enregistrement est en train d'être modifié.")
      Fin de si
   Fin de si
Fin de si
```

Variables et ensembles système

La variable système OK prend la valeur 1 si l'enregistrement est validé et 0 s'il est annulé.

Référence

CREER ENREGISTREMENT, MODIFIER ENREGISTREMENT, NE PAS VALIDER, STOCKER ENREGISTREMENT, VALIDER.

MODIFIER ENREGISTREMENT ({table}{; }{*})

Paramètre	Type		Description
table	Table	→	Table dans laquelle modifier des données ou Table par défaut si ce paramètre est omis
*		→	Cacher les barres de défilement

Description

La commande MODIFIER ENREGISTREMENT permet à l'utilisateur de modifier l'enregistrement courant de table, ou de la table par défaut si ce paramètre est omis. MODIFIER ENREGISTREMENT charge depuis le disque l'enregistrement courant pour le process en cours (s'il n'est pas déjà chargé par un autre utilisateur/process) et l'affiche dans le formulaire entrée courant. S'il n'y a pas d'enregistrement courant, MODIFIER ENREGISTREMENT ne fait rien. MODIFIER ENREGISTREMENT ne change pas la sélection courante.
Le formulaire est affiché dans la fenêtre se trouvant au premier plan du process. Elle comporte des barres de défilement et une case de contrôle de taille. Si vous passez le paramètre optionnel astérisque (*), ces éléments n'apparaîtront pas.

Pour que vous puissiez utiliser MODIFIER ENREGISTREMENT, l'enregistrement courant doit être en Lecture/écriture et ne doit pas être verrouillé.

Si le formulaire comporte des boutons de navigation parmi les enregistrements de la sélection, ils restent utilisables, ce qui permet à l'utilisateur de modifier des enregistrements puis de se déplacer pour en modifier d'autres.

L'enregistrement est sauvegardé si l'utilisateur clique sur un bouton du type Valider ou appuie sur la touche **Entrée**, ou encore si la commande VALIDER est exécutée.

L'enregistrement n'est pas sauvegardé si l'utilisateur clique sur un bouton du type Annuler ou appuie sur la touche d'annulation (**Echap** sous Windows, **Esc** sous MacOS), ou encore si la commande NE PAS VALIDER est exécutée.

Après un appel à MODIFIER ENREGISTREMENT, la variable système OK prend la valeur 1 si l'enregistrement est validé, et 0 lorsqu'il est annulé.

Note : Même lorsqu'il est annulé, l'enregistrement reste en mémoire et peut être sauvegardé par la commande STOCKER ENREGISTREMENT si celle-ci est appelée avant que le pointeur d'enregistrement courant ne soit modifié.

Dans le cadre d'un MODIFIER ENREGISTREMENT, si l'utilisateur n'effectue aucune modification dans l'enregistrement et le valide, l'enregistrement ne sera pas considéré comme modifié et ne sera pas sauvegardé une nouvelle fois. Les actions telles que le changement de la valeur d'une variable, la sélection de cases à cocher ou de boutons radio ne sont pas qualifiées de modifications. Seule la modification de la valeur d'un champ, par le biais d'une saisie manuelle ou d'une méthode, provoque une nouvelle sauvegarde de l'enregistrement.

Exemples

Reportez-vous au second exemple de la commande AJOUTER ENREGISTREMENT.

Variables et ensembles système

La variable système OK prend la valeur 1 lorsque l'enregistrement est validé et 0 lorsqu'il est annulé. OK ne prend une valeur qu'après que l'enregistrement ait été effectivement validé ou annulé.

Référence

AJOUTER ENREGISTREMENT, Enregistrement modifie, Enregistrement verrouille, LECTURE ECRITURE, LIBERER ENREGISTREMENT.

AJOUTER SOUS ENREGISTREMENT

Saisie

version 5

AJOUTER SOUS ENREGISTREMENT (sousTable; formulaire{; *})

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table à utiliser pour la saisie des données
formulaire	Alpha	→	Formulaire à utiliser
*	*	→	Masquer les barres de défilement

Description

AJOUTER SOUS ENREGISTREMENT permet d'ajouter un nouveau sous-enregistrement à sousTable, en utilisant formulaire. AJOUTER SOUS ENREGISTREMENT crée un nouveau sous-enregistrement en mémoire, en fait le sous-enregistrement courant et affiche formulaire. Il doit exister un enregistrement courant pour la table parente. S'il n'existe pas d'enregistrement de la table parente dans le process, AJOUTER SOUS ENREGISTREMENT ne fait rien. Le formulaire doit appartenir à sousTable.

Le sous-enregistrement reste en mémoire et sera sauvegardé si l'utilisateur clique sur un bouton de validation, appuie sur la touche **Entrée** ou si la commande VALIDER est exécutée. Dès que le sous-enregistrement a été ajouté ou modifié, l'enregistrement parent doit être sauvegardé pour que le sous-enregistrement soit sauvegardé.

Le sous-enregistrement n'est pas sauvegardé si l'utilisateur clique sur un bouton d'annulation, appuie sur les touches d'annulation (**Echap** sous Windows, **Esc** sous MacOS), ou si la commande NE PAS VALIDER est exécutée.

Après un appel à AJOUTER SOUS ENREGISTREMENT, la variable système OK prend la valeur 1 si le sous-enregistrement a été validé, sinon elle prend la valeur 0.

Le formulaire est affiché dans la fenêtre de premier plan du process, avec des barres de défilement et une case de redimensionnement. Si vous spécifiez le paramètre optionnel astérisque (*), la fenêtre sera dessinée sans les barres de défilement ni la case de redimensionnement.

Exemple

L'exemple suivant fait partie d'une méthode. Ces lignes de code ajoutent un sous-enregistrement pour un nouvel enfant dans l'enregistrement d'un employé. Les données pour l'enfant sont stockées dans la sous-table [Employés]Enfants. Notez que l'enregistrement de la table [Employés] doit être sauvegardé pour que le sous-enregistrement le soit également :

```
⇒   AJOUTER SOUS ENREGISTREMENT ([Employés]Enfants; "AjouterEnfant")
    Si (OK = 1)  ` Si l'utilisateur a validé le sous-enregistrement
       STOCKER ENREGISTREMENT ([Employés])  ` Stocker l'enregistrement parent
    Fin de si
```

Variables et ensembles système

Si le sous-enregistrement est validé, la variable système OK prend la valeur 1, sinon elle prend la valeur 0.

Référence

MODIFIER SOUS ENREGISTREMENT, NE PAS VALIDER, STOCKER ENREGISTREMENT, VALIDER.

MODIFIER SOUS ENREGISTREMENT

Saisie

version 5

MODIFIER SOUS ENREGISTREMENT (sousTable; formulaire{; *})

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table à utiliser pour la saisie de données
formulaire		→	Formulaire à utiliser pour la saisie
*		→	Cacher les barres de défilement

Description

La commande MODIFIER SOUS ENREGISTREMENT affiche le sous-enregistrement courant de sousTable dans le formulaire en mode modification.

La table parente doit disposer d'un enregistrement courant. S'il n'y a pas d'enregistrement parent courant pour le process, MODIFIER SOUS ENREGISTREMENT ne fait rien. De plus, s'il n'y a pas de sous-enregistrement courant, MODIFIER SOUS ENREGISTREMENT ne fait rien non plus.

Le sous-enregistrement reste en mémoire et sera sauvegardé si l'utilisateur clique sur le bouton OK, appuie sur la touche Entrée ou si la commande VALIDER est exécutée. Dès que le sous-enregistrement a été ajouté ou modifié, l'enregistrement parent doit être sauvegardé pour que le sous-enregistrement le soit également.

Le sous-enregistrement n'est pas sauvegardé si l'utilisateur clique sur le bouton **Annuler**, appuie sur la touche **Echap** (Windows) ou **Esc** (MacOS), ou si la commande NE PAS VALIDER est exécutée.

Après un appel à MODIFIER SOUS ENREGISTREMENT, si le sous-enregistrement est validé, la variable système OK prend la valeur 1, sinon elle prend la valeur 0.

Le formulaire est affiché dans la fenêtre de premier plan du process avec des barres de défilement et une case de redimensionnement. Si vous spécifiez l'astérisque optionnel (*), la fenêtre sera dessinée sans les barres de défilement ni la case de redimensionnement.

Variables et ensembles système

Si le sous-enregistrement est validé, la variable système OK prend la valeur 1, sinon elle prend la valeur 0.

Référence

AJOUTER SOUS ENREGISTREMENT, NE PAS VALIDER, STOCKER ENREGISTREMENT, VALIDER.

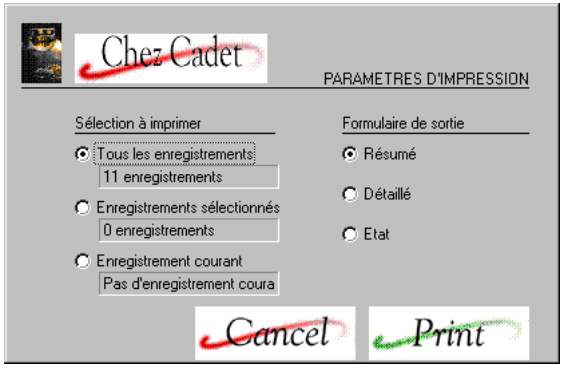
DIALOGUE ({table; }formulaire)

Paramètre	Type		Description
table	Table	→	Table à laquelle appartient le formulaire ou Table par défaut si ce paramètre est omis
formulaire	Formulaire	→	Formulaire à afficher comme dialogue

Description

La commande DIALOGUE présente le formulaire à l'utilisateur. Cette commande est souvent utilisée pour récupérer dans des variables des données fournies par l'utilisateur, ou pour lui présenter différentes informations, comme un choix d'options pour effectuer une opération.

Il est courant d'afficher le formulaire dans une fenêtre modale créée à l'aide de la commande Creer fenetre.
Voici un exemple typique de boîte de dialogue pouvant être affichée avec la commande DIALOGUE :



Dans ce type de boîte de dialogue, la saisie de valeurs ne peut s'effectuer que par l'intermédiaire de variables. Des champs peuvent être affichés avec leurs valeurs courantes mais ils ne sont pas saisissables.

Astuce : Vous pouvez simuler l'affichage d'une boîte de dialogue avec la commande AJOUTER ENREGISTREMENT, si vous souhaitez bénéficier des possibilités issues de la saisie de données dans des champs. Dans ce cas, lorsque le formulaire est validé, un enregistrement est créé dans la table.

Astuce : A l'inverse, il est possible de créer ou de modifier des valeurs dans les enregistrements par l'intermédiaire de la commande DIALOGUE. Dans ce cas, c'est à vous de créer et de sauvegarder par programmation les enregistrements. DIALOGUE ne manipule pas les enregistrements.

Utilisez DIALOGUE plutôt que CONFIRMER, ALERTE ou Demander lorsque les informations à afficher ou à recueillir sont plus complexes que celles que peuvent gérer ces trois autres commandes.

A la différence d'AJOUTER ENREGISTREMENT et de MODIFIER ENREGISTREMENT, DIALOGUE n'utilise pas le formulaire entrée courant. Vous devez spécifier, dans le paramètre formulaire, le formulaire à utiliser. De même, aucun ensemble de boutons n'est placé par défaut s'ils sont omis dans le formulaire. Seuls des boutons **OK** et **Annuler** sont créés automatiquement. Ils sont supprimés si le formulaire contient des boutons personnalisés.

Le dialogue est validé si l'utilisateur clique sur le bouton de validation ou appuie sur la touche **Entrée**, ou si la commande VALIDER est exécutée.
Le dialogue est annulé si l'utilisateur clique sur le bouton d'annulation, appuie sur la touche d'annulation (**Esc**), ou si la commande NE PAS VALIDER est exécutée.

Après un appel à DIALOGUE, la variable système OK prend la valeur 1 si le dialogue est validé, et 0 sinon.

Exemple

L'exemple suivant illustre l'utilisation de la commande DIALOGUE pour spécifier des critères de recherche. Un formulaire personnalisé contenant les variables vNom et vPays permet à l'utilisateur de saisir ses critères :

```
      Creer fenetre (10; 40; 370; 220) ` Créer une fenêtre modale
⇒     DIALOGUE ([Société]; "Form Recherche") ` Afficher le dialogue de recherche
      FERMER FENETRE ` Nous n'avons plus besoin de la fenêtre
      Si (OK = 1) ` Si le dialogue est validé
         CHERCHER ([Société]; [Société]Nom = vNom; *)
         CHERCHER (& [Société]Payst = vPays)
      Fin de si
```

Variables et ensembles système

Si l'utilisateur valide le dialogue, la variable système OK prend la valeur 1, si le dialogue est annulé OK prend la valeur 0.

Référence

AJOUTER ENREGISTREMENT, Creer fenetre, NE PAS VALIDER, VALIDER.

Modifie

Saisie

version 5

Modifie (champ) → Booléen

Paramètre	Type		Description
champ	Champ	→	Champ dont vous voulez tester la modification
Résultat	Booléen	←	Vrai si une nouvelle valeur a été assignée au champ, sinon Faux

Description

Modifie retourne Vrai si une valeur a été assignée par programmation au champ champ ou s'il a été modifié lors de la saisie de données. La commande LIRE VARIABLES ne fonctionne que lorsqu'elle est appelée dans le cadre d'une méthode formulaire (ou d'une sous-méthode appelée par la méthode formulaire).

Dans le cas de la saisie de données, un champ est considéré comme modifié à partir du moment où un utilisateur l'édite (et change ou non sa valeur originale) puis le quitte pour un autre champ ou pour cliquer sur un objet de formulaire. Notez que le fait qu'un utilisateur active puis quitte un champ à l'aide de la touche Tabulation ne suffit pas en soi à ce que Modifie retourne Vrai. Le champ doit avoir été réellement modifié pour que Modifie retourne Vrai.

Dans le cas de l'exécution d'une méthode, un champ est considéré comme modifié si une valeur lui a été assignée (différente ou non de sa valeur précédente).

Note : La commande LIRE VARIABLES retourne toujours Vrai après l'exécution des commandes EMPILER ENREGISTREMENT et DEPILER ENREGISTREMENT.

Dans tous les cas, pour savoir si la valeur d'un champ a été effectivement modifiée, utilisez la commande Ancien.

Note : Bien que la fonction Modifie puisse être appliquée à tout type de champ, si vous l'utilisez conjointement avec la fonction Ancien, vous devez dans ce cas tenir compte des restrictions liées à cette fonction. Reportez-vous à la description de Ancien.

Pendant la saisie de données, il est généralement plus pratique d'effectuer des opérations dans des méthodes objet que d'utiliser la fonction Modifie dans des méthodes formulaires. Comme une méthode objet reçoit l'événement Sur données modifiées à chaque fois qu'un champ est modifié, utiliser une telle méthode équivaut à appeler Modifie dans une méthode formulaire.

Exemples

(1) L'exemple suivant teste si le champ [Commandes]Quantité ou le champ [Commandes]Prix a été modifié. Si c'est le cas, le champ [Commandes]Total est recalculé :

```
⇒   Si ((Modifie ([Commandes]Quantité) | (Modifie ([Commandes]Prix))
        [Commandes]Total := [Commandes]Quantité * [Commandes]Prix
    Fin de si
```

Notez que le même résultat aurait pu être obtenu en utilisant la seconde ligne de cette méthode en tant que méthode objet des champs [Commandes]Quantité et [Commandes]Prix.

(2) Vous sélectionnez un enregistrement de la table [uneTable], puis vous appelez plusieurs sous-routines qui sont susceptibles de modifier le champ [uneTable]Champ important mais sans provoquer de sauvegarde de l'enregistrement. A la fin de la méthode principale, vous pouvez utiliser la commande Modifie pour déterminer si vous devez stocker l'enregistrement :

```
    ` L'enregistrement a été sélectionné comme enregistrement courant
    ` Puis vous effectuez des actions à l'aide des sous-routines
FAIRE QUELQUE CHOSE
FAIRE AUTRE CHOSE
NE PAS OUBLIER DE FAIRE CA
    ` ...
    ` Enfin, vous testez le champ pour déterminer s'il faut stocker l'enregistrement
⇒   Si (Modifie([uneTable]Champ important))
        STOCKER ENREGISTREMENT([uneTable])
    Fin de si
```

Référence

Ancien.

Ancien

Saisie

version 5

Ancien (champ) → Expression

Paramètre	Type		Description
champ	Champ	→	Champ dont vous voulez obtenir l'ancienne valeur
Résultat	Expression	←	Valeur originale de champ

Description

La commande Ancien retourne la valeur qui était stockée dans champ avant qu'il n'ait été modifié par programmation ou pendant la saisie de données.

A chaque fois que vous changez d'enregistrement courant pour une table, 4D crée et maintient en mémoire un double de l'"image" du nouvel enregistrement courant au moment où il est chargé (pour des raisons d'optimisation, ce fonctionnement ne s'applique pas aux champs de type Texte, Image et BLOB). Lorsque vous modifiez un enregistrement, vous travaillez avec l'image réelle de l'enregistrement, et non son double. Ce double est effacé lorsque que vous changez à nouveau d'enregistrement courant.

Ancien retourne la valeur de champ telle qu'elle est stockée dans le double de l'enregistrement. Autrement dit, pour un enregistrement existant, Ancien retourne la valeur du champ telle qu'elle avait été sauvegardée sur disque. Pour un enregistrement qui vient d'être créé, Ancien retourne la valeur vide par défaut correspondant au type de champ. Par exemple, si champ est de type Alpha, Ancien retourne une chaîne vide. Si champ est de type numérique, Ancien retourne zéro (0), etc.

Ancien fonctionne avec champ de la même manière, que le champ ait été modifié par programmation ou suite à des modifications effectuées par un utilisateur.

Ancien ne peut pas être appliquée aux champs de type Texte, Image ou BLOB. La fonction accepte tous les autres types de champs, y compris les sous-champs ; à noter cependant que son utilisation avec un champ de type Sous-table n'a pas de sens.

Pour restaurer la valeur originale d'un champ, assignez-lui la valeur retournée par Ancien.

Référence

Modifie.

47

Sélections

TOUT SELECTIONNER

TOUT SELECTIONNER {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle vous voulez sélectionner tous les enregistrements ou Table par défaut si ce paramètre est omis

Description

La commande TOUT SELECTIONNER sélectionne tous les enregistrements de table pour le process courant. TOUT SELECTIONNER fait du premier enregistrement de la sélection l'enregistrement courant et le charge en mémoire. TOUT SELECTIONNER retourne les enregistrements dans l'ordre par défaut, qui est l'ordre dans lequel ils ont été stockés sur le disque.

Exemple

L'exemple suivant affiche tous les enregistrements de la table [Personnes] :

```
⇒ TOUT SELECTIONNER([Personnes]) ` Sélection de tous les enregistrements de la table
  VISUALISER SELECTION ([Personnes]) `Affichage dans le formulaire sortie
```

Référence

CHERCHER, Enregistrements dans table, Enregistrements trouves, MODIFIER SELECTION, TRIER, VISUALISER SELECTION.

Enregistrements trouves {(table)} → Numérique

Paramètre	Type		Description
table	Table	→	Table dont vous souhaitez connaître le nombre d'enregistrements de la sélection courante ou Table par défaut si ce paramètre est omis
Résultat	Numérique	←	Nombre d'enregistrements dans la sélection courante de table

Description

Enregistrements trouves retourne le nombre d'enregistrements constituant la sélection courante de table (par opposition, Enregistrements dans table retourne le nombre total d'enregistrements d'une table).

Exemple

L'exemple suivant propose une technique de boucle couramment utilisée pour se déplacer parmi les enregistrements de la sélection courante. La même opération peut être réalisée à l'aide de la commande APPLIQUER A SELECTION :

```
      ` Départ sur le premier enregistrement de la sélection
   DEBUT SELECTION ([Personnes])
⇒  Boucle ($VElEnreg; 1; Enregistrements trouves ([Personnes]))
      Faire quelque chose    ` Réaliser une opération avec l'enregistrement
      ENREGISTREMENT SUIVANT ([Personnes])    ` Passage à l'enregistrement suivant
   Fin de boucle
```

Référence

Enregistrements dans table.

SUPPRIMER SELECTION {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle supprimer la sélection courante ou Table par défaut si ce paramètre est omis

Description

La commande SUPPRIMER SELECTION supprime la sélection courante d'enregistrements de table. Si la sélection courante est vide, SUPPRIMER SELECTION ne fait rien. Après la suppression des enregistrements, la sélection courante est vide. Les enregistrements supprimés pendant une transaction sont verrouillés pour les autres utilisateurs et/ou process jusqu'à ce que la transaction soit validée ou annulée.

Attention : La suppression d'une sélection d'enregistrements est une opération définitive. Elle ne peut être annulée par la suite.

L'option **Suppression physique** de la boîte de dialogue de définition des Propriétés des tables vous permet d'augmenter la vitesse des suppressions lors de l'utilisation de SUPPRIMER SELECTION.

Exemple

(1) L'exemple suivant affiche tous les enregistrements de la table [Personnes] et permet à l'utilisateur de sélectionner ceux qu'il souhaite effacer. L'exemple est en deux parties. La première est la méthode affichant les enregistrements. La seconde est la méthode objet d'un bouton 'Supprimer'. Voici la première méthode :

```
TOUT SELECTIONNER ([Personnes]) ` Sélection de tous les enregistrements
   ` Définition du formulaire listant les enregistrements
FORMULAIRE SORTIE ([Personnes]; "FormSortie")
VISUALISER SELECTION ([Personnes]) ` Affichage de tous les enregistrements
```

Voici la méthode objet du bouton Supprimer, apparaissant dans le pied de page du formulaire sortie. La méthode utilise les enregistrements sélectionnés par l'utilisateur (l'ensemble système UserSet) pour effacer la sélection (notez que si l'utilisateur ne sélectionne aucun enregistrement, SUPPRIMER SELECTION ne fait rien) :

```
      ` Demander confirmation que l'utilisateur veut réellement supprimer les
                                                                    enregistrements
   CONFIRMER ("Vous avez sélectionné "+Chaine(Enregistrements dans
                                          ensemble("UserSet"))+" enregistrements à supprimer.
                                    "+Caractere(13)+"Cliquez sur OK pour confirmer l'opération.")
   Si(OK=1)
      UTILISER ENSEMBLE("UserSet") ` Utiliser l'ensemble défini par l'utilisateur
⇒     SUPPRIMER SELECTION([Personnes]) ` Supprimer la sélection d'enregistrements
   Fin de si
   TOUT SELECTIONNER([Personnes]) ` Sélection de tous les enregistrements
```

(2) Lorsqu'un SUPPRIMER SELECTION rencontre un enregistrement verrouillé, celui-ci n'est pas supprimé. Tous les enregistrements verrouillés sont placés dans un ensemble système nommé LockedSet. Après l'exécution de SUPPRIMER SELECTION, vous pouvez tester cet ensemble afin de vérifier si des enregistrements étaient verrouillés. La boucle suivante s'exécutera jusqu'à ce que tous les enregistrements aient été supprimés.

```
   Repeter  ` Répéter pour chaque enregistrement verrouillé
⇒     SUPPRIMER SELECTION ([CetteTable])
         ` Si des enregistrements sont verrouillés
      Si (Enregistrements dans ensemble("LockedSet")#0)
         UTILISER ENSEMBLE("LockedSet")    ` Sélectionner les enregistrements verrouillés
      Fin de si
   Jusque (Enregistrements dans ensemble("LockedSet")=0)   ` Jusqu'à ce qu'il n'y en ait
                                                                    plus
```

Référence

MODIFIER SELECTION, Présentation des ensembles, Verrouillage d'enregistrements, VISUALISER SELECTION.

Numero dans selection

Numero dans selection {(table)} → Numérique

Paramètre	Type		Description
table	Table	→	Table de laquelle vous voulez obtenir le numéro de l'enregistrement courant dans la sélection
Résultat	Numérique	←	Numéro dans la sélection

Description

Numero dans selection retourne la position de l'enregistrement courant dans la sélection courante de table.

Si la sélection est non vide et si l'enregistrement courant en fait partie, Numero dans selection retourne une valeur comprise entre 1 et Enregistrements trouves. Si la sélection est vide ou s'il n'y a pas d'enregistrement courant, Numero dans selection retourne 0.

Le numéro de l'enregistrement dans la sélection est différent du numéro retourné par Numero enregistrement (Numero enregistrement retourne le numéro physique de l'enregistrement dans la table). Le numéro de l'enregistrement dans la sélection dépend de la sélection courante.

Pour plus de précisions sur les numéros d'enregistrements, reportez-vous à la section A propos des numéros d'enregistrements.

Exemple

L'exemple suivant stocke le numéro de l'enregistrement courant de la sélection dans une variable :

```
      ` Obtenir le numéro de l'enregistrement dans la sélection
⇒     NumEnrCourant := Numero dans selection ([Personnes])
```

Référence

A propos des numéros d'enregistrements, ALLER DANS SELECTION, Enregistrements trouves.

ALLER DANS SELECTION ({table; }position)

Paramètre	Type		Description
table	Table	→	Table dans laquelle aller à l'enregistrement spécifié ou Table par défaut si ce paramètre est omis
position	Numérique	→	Position de l'enregistrement dans la sélection

Description

La commande ALLER DANS SELECTION fait de l'enregistrement spécifié parmi la sélection courante de table l'enregistrement courant. La sélection courante n'est pas modifiée. Le paramètre position n'est pas équivalent au numéro retourné par Numero enregistrement. Ce paramètre représente la position de l'enregistrement au sein de la sélection courante. Cette position dépend de la manière dont la sélection a été créée et si elle a été triée.

Pour plus de précisions sur les numéros d'enregistrements, reportez-vous à la section A propos des numéros d'enregistrements.

S'il n'y a aucun enregistrement dans la sélection courante ou si position n'est pas dans la sélection, ALLER DANS SELECTION ne fait rien.

Exemple

L'exemple suivant charge les valeurs du champ [Personnes]Nom dans le tableau taNoms. Un tableau d'entiers longs, numEnr, est rempli avec des numéros qui représenteront ceux des enregistrements sélectionnés. Les deux tableaux sont alors triés :

```
   ` Créer ici la sélection de la table [Personnes]
   `...
   ` Récupérer les noms
SELECTION VERS TABLEAU ([Personnes]Nom; taNoms)
   ` Créer un tableau pour les numéros d'enregistrements sélectionnés
$vELNbEnrgs:=Taille tableau (taNoms)
TABLEAU ENTIER LONG (numEnr; $vELNbEnrgs)
Boucle ($Enrg; 1;$vELNbEnrgs)  ` Remplir le tableau avec ces numéros
   numEnr{$Enrg} := $Enrg
Fin de boucle
   ` Trier les deux tableaux par ordre alphabétique
TRIER TABLEAU (taNoms; numEnr; >)
```

Si le tableau taNoms est affiché dans une zone de défilement, l'utilisateur peut cliquer sur l'un des éléments. Comme les deux tableaux ont été triés de manière synchronisée, tout élément de numEnr fournit le numéro de l'enregistrement sélectionné pour lequel le nom a été stocké dans l'élément de taNoms correspondant.

La méthode objet de la zone de défilement taNoms suivante sélectionne le bon enregistrement dans la sélection de [Personnes] en fonction de ce que l'utilisateur a choisi dans la zone de défilement.

```
      Au cas ou
         : (Evenement formulaire=Sur clic souris)
            Si (taNoms#0)
⇒              ALLER DANS SELECTION (numEnr{taNoms})
            Fin de si
       Fin de cas
```

Référence

Numero dans selection.

DEBUT SELECTION

DEBUT SELECTION {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle charger le premier enregistrement de la sélection courante ou Table par défaut si ce paramètre est omis

Description

DEBUT SELECTION charge en mémoire le premier enregistrement de la sélection courante de table et en fait l'enregistrement courant. Toutes les commandes de recherche, de sélection et de tri font également du premier enregistement l'enregistrement courant. Si la sélection courante est vide, DEBUT SELECTION ne fait rien.

Cette commande est principalement utilisée après un appel à UTILISER ENSEMBLE, pour débuter une boucle dans la sélection d'enregistrements à partir du premier enregistrement. Cependant, il est tout à fait envisageable de l'appeler depuis une sous-routine lorsque vous souhaitez vous assurer que l'enregistrement est bien le premier.

Exemple

L'exemple suivant charge le premier enregistrement de la table [Clients] :

```
⇒ DEBUT SELECTION ([Clients])
```

Référence

ALLER A DERNIER ENREGISTREMENT, Avant selection, ENREGISTREMENT PRECEDENT, ENREGISTREMENT SUIVANT, Fin de selection.

ENREGISTREMENT SUIVANT {(table)}

Paramètre	Type		Description
table	Table	→	Table dans laquelle se placer sur l'enregistrement suivant ou Table par défaut si ce paramètre est omis

Description

La commande ENREGISTREMENT SUIVANT place le pointeur d'enregistrement courant sur l'enregistrement suivant dans la sélection courante de table pour le process courant. Si la sélection courante est vide, ou si Avant selection ou Fin de selection retourne Vrai, ENREGISTREMENT SUIVANT ne fait rien.

Si ENREGISTREMENT SUIVANT place le pointeur d'enregistrement courant après la fin de la sélection courante, Fin de selection retourne Vrai, et il n'y a alors plus d'enregistrement courant. Lorsque Fin de selection retourne Vrai, utilisez les commandes DEBUT SELECTION, ALLER A DERNIER ENREGISTREMENT ou ALLER DANS SELECTION pour replacer le pointeur d'enregistrement courant dans la sélection courante.

Exemple

Reportez-vous à l'exemple de la commande VISUALISER SELECTION.

Référence

ALLER A DERNIER ENREGISTREMENT, Avant selection, DEBUT SELECTION, ENREGISTREMENT PRECEDENT, Fin de selection.

ALLER A DERNIER ENREGISTREMENT {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle vous voulez aller au dernier enregistrement ou Table par défaut si ce paramètre est omis

Description

ALLER A DERNIER ENREGISTREMENT désigne le dernier enregistrement de la sélection de table comme enregistrement courant et le charge en mémoire. Si la sélection est vide, ALLER A DERNIER ENREGISTREMENT ne fait rien.

Exemple

L'exemple suivant désigne le dernier enregistrement de la table [Contacts] comme enregistrement courant :

```
⇒ ALLER A DERNIER ENREGISTREMENT ([Contacts])
```

Référence

Avant selection, DEBUT SELECTION, ENREGISTREMENT PRECEDENT, ENREGISTREMENT SUIVANT, Fin de selection.

ENREGISTREMENT PRECEDENT

version 5

ENREGISTREMENT PRECEDENT {(table)}

Paramètre	Type		Description
table	Table	→	Table dans laquelle se placer sur l'enregistrement précédent de la sélection courante ou Table par défaut si ce paramètre est omis

Description

ENREGISTREMENT PRECEDENT place le pointeur d'enregistrement courant sur l'enregistrement précédent dans la sélection courante de table pour le process courant. Si la sélection courante est vide, ou si Avant selection ou Fin de selection renvoie Vrai, ENREGISTREMENT PRECEDENT ne fait rien.

Si ENREGISTREMENT PRECEDENT place le pointeur d'enregistrement courant avant la sélection courante, Avant selection retourne Vrai, et il n'y a plus d'enregistrement courant. Dans ce cas, utilisez les commandes DEBUT SELECTION, ALLER A DERNIER ENREGISTREMENT ou ALLER DANS SELECTION pour replacer le pointeur d'enregistrement courant dans la sélection courante.

Référence

ALLER A DERNIER ENREGISTREMENT, Avant selection, DEBUT SELECTION, ENREGISTREMENT SUIVANT, Fin de selection.

Avant selection

Avant selection {(table)} → Booléen

Paramètre	Type		Description
table	Table	→	Table pour laquelle vous testez si le pointeur se trouve avant la sélection
Résultat	Booléen	←	Avant sélection (Vrai) sinon (Faux)

Description

La fonction Avant selection retourne Vrai lorsque le pointeur d'enregistrement courant se trouve avant le premier enregistrement de la sélection courante de table. Avant selection est généralement utilisée pour vérifier si la commande ENREGISTREMENT PRECEDENT a déplacé le pointeur d'enregistrement courant avant le premier enregistrement. Si la sélection courante est vide, Avant selection retourne Vrai.

Pour replacer le pointeur d'enregistrement courant dans la sélection courante, utilisez les commandes DEBUT SELECTION, ALLER A DERNIER ENREGISTREMENT ou ALLER DANS SELECTION. ENREGISTREMENT SUIVANT ne replace pas le pointeur d'enregistrement courant dans la sélection courante.

Avant selection retourne Vrai dans l'en-tête lorsqu'un état est en cours d'impression à l'aide de la commande IMPRIMER SELECTION ou à partir de la commande de menu Imprimer. Vous pouvez utiliser le code suivant pour tester le premier en-tête et imprimer un en-tête spécial pour la première page :

```
    ` Méthode d'un formulaire sortie utilisé pour un état
$vpFormTable:=Table du formulaire courant
Au cas ou
    ` ...
  : (Evenement formulaire=Sur entête)
      ` La zone en-tête va être imprimée
    Au cas ou
⇒     : (Avant selection($vpFormTable->))
        ` Le code pour la première rupture d'en-tête doit être placé ici
        ` ...
    Fin de cas
Fin de cas
```

Exemple

La méthode formulaire suivante est utilisée pendant l'impression d'un état. Elle définit une variable vTitre à imprimer dans la zone d'en-tête sur la première page :

```
      ` Méthode formulaire [Finances];"Tableau"
   Au cas ou
      ` ...
      : (Evenement formulaire=Sur entête)
         ` La zone en-tête va être imprimée
         Au cas ou
⇒           : (Avant selection([Finances]))
               vTitre := "Etat des finances pour 1997"  ` Titre de la première page
            Sinon
               vTitre := ""  ` Effacer le titre pour les autres pages
         Fin de cas
   Fin de cas
```

Référence

DEBUT SELECTION, ENREGISTREMENT PRECEDENT, Evenement formulaire, Fin de selection, IMPRIMER SELECTION.

Fin de selection {(table)} → Booléen

Paramètre	Type		Description
table	Table	→	Table pour laquelle tester si le pointeur d'enregistrement courant est au-delà du dernier enregistrement de la sélection courante ou Table par défaut si ce paramètre est omis
Résultat	Booléen	←	Oui (Vrai), Non (Faux)

Description

La fonction Fin de selection retourne Vrai lorsque le pointeur de l'enregistrement courant se trouve après le dernier enregistrement de la sélection courante de table. Fin de selection est généralement utilisée pour tester si l'appel à la commande ENREGISTREMENT SUIVANT place ou non le pointeur d'enregistrement courant derrière le dernier enregistrement de la sélection. Si la sélection courante est vide, Fin de selection retourne Vrai.

Pour replacer le pointeur d'enregistrement courant dans la sélection, utilisez les commandes ALLER A DERNIER ENREGISTREMENT, DEBUT SELECTION ou ALLER DANS SELECTION. ENREGISTREMENT PRECEDENT ne replace pas le pointeur dans la sélection.

Fin de selection retourne également Vrai lors de l'impression du dernier pied de page d'un état, déclenchée par la commande IMPRIMER SELECTION ou le menu Imprimer. Vous pouvez utiliser l'instruction suivante pour intercepter le dernier pied de page et insérer une mention particulière :

```
    ` Méthode d'un formulaire sortie utilisé pour imprimer un état
$vpFormTable:=Table du formulaire courant
Au cas ou
      ` ...
   : (Evenement formulaire=Sur impression pied de page)
         ` Un pied
⇒        Si (Fin de selection($vpFormTable->))
            ` Le code pour le dernier pied de page doit être placé ici
         Sinon
            ` Le code pour le pied de page doit être placé ici
         Fin de si
Fin de cas
```

Exemple

La méthode formulaire de l'exemple suivant est utilisée lors de l'impression d'un état. Elle crée la variable VPied, à imprimer dans le pied de page de la dernière page :

```
      ` Méthode formulaire [Finances];"Tableau"
   Au cas ou
      ` ...
      : (Evenement formulaire=Sur impression pied de page)
⇒        Si (Fin de selection([Finances]))
            VPied:= "©1997 SARL Dupont"  ` Définir le pied de page de la dernière page
         Sinon
            VPied:= ""  ` Effacer le pied de page pour toutes les autres pages
         Fin de si
   Fin de cas
```

Référence

ALLER A DERNIER ENREGISTREMENT, Avant selection, ENREGISTREMENT SUIVANT, Evenement formulaire, IMPRIMER SELECTION.

VISUALISER SELECTION

Sélections

version 6 (Modifiée)

VISUALISER SELECTION ({table}{; *}; *)

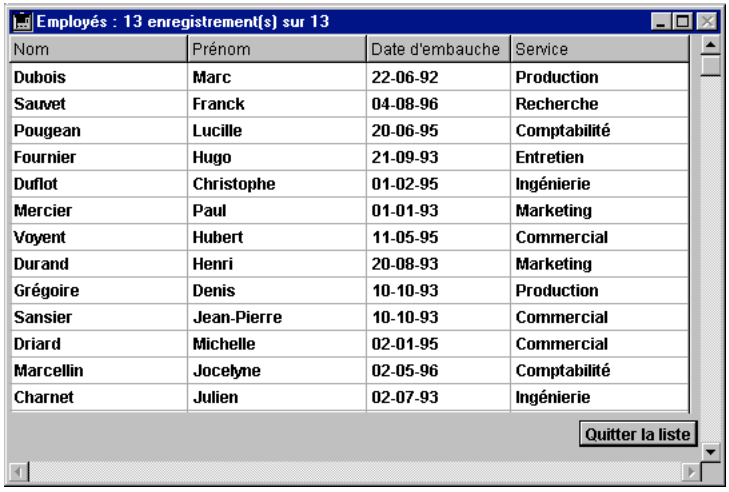
Paramètre	Type		Description
table	Table	→	Table à laquelle appartient la sélection ou Table par défaut si ce paramètre est omis
*		→	Utiliser le formulaire sortie en cas de sélection d'un seul enregistrement et masquer les barres de défilement dans le formulaire entrée
*		→	Afficher les barres de défilement dans le formulaire entrée (= annuler le second effet du premier paramètre *)

Description

VISUALISER SELECTION affiche, pour le process en cours, la sélection courante de table dans le formulaire sortie courant. Les enregistrements sont affichés sous la forme d'une liste que l'on peut faire défiler, semblable à celle du mode Utilisation. Lorsque l'utilisateur double-clique sur un enregistrement, celui-ci s'affiche dans le formulaire entrée. La liste est placée dans la fenêtre de premier plan.

Si vous souhaitez afficher une sélection et pouvoir également modifier un enregistrement une fois que vous avez double-cliqué dessus (comme vous le faites dans la fenêtre du mode Utilisation) utilisez MODIFIER SELECTION au lieu de VISUALISER SELECTION. Toutes les explications suivantes s'appliquent à ces deux commandes, hormis la possibilité de modifier des enregistrements.

L'écran suivant présente un formulaire sortie affiché par la commande VISUALISER SELECTION.



Après qu'un VISUALISER SELECTION ait été exécuté, il n'y a plus d'enregistrement courant. Vous devez utiliser une commande telle que DEBUT SELECTION ou ALLER A DERNIER ENREGISTREMENT pour en récupérer un.

VISUALISER SELECTION ne permet pas à l'utilisateur de modifier un enregistrement affiché dans un formulaire entrée. En revanche, MODIFIER SELECTION permet cette opération.

Lorsque la sélection ne contient qu'un enregistrement, et que le premier paramètre optionnel * n'est pas passé, l'enregistrement s'affichera directement dans le formulaire entrée. Si le premier paramètre optionnel * est spécifié, l'enregistrement unique sera affiché dans le formulaire sortie. Si le premier paramètre optionnel * est spécifié et que l'utilisateur affiche l'enregistrement dans le formulaire entrée en double-cliquant dessus, les barres de défilement du formulaire seront masquées. Pour annuler ce second effet du premier paramètre optionnel *, passez le second paramètre optionnel *.

Un bouton libellé 'Quitter la liste' est automatiquement placé en bas de la liste. Un clic sur ce bouton provoque la fin de l'exécution de la commande VISUALISER SELECTION. L'ajout de toute variable ou de tout objet actif dans le formulaire entraîne la disparition de ce bouton. Vous pouvez placer des boutons personnalisés dans la zone de Pied de page du formulaire sortie. Vous pouvez utiliser des boutons automatiques **Valider** ou **Annuler** permettant de sortir de la liste ou utiliser une méthode objet qui appelle les commandes VALIDER ou NE PAS VALIDER.

L'utilisateur peut faire défiler la sélection et cliquer sur un enregistrement pour le sélectionner. S'il clique ensuite sur un autre enregistrement, le premier se désélectionne. Il est cependant possible de sélectionner plusieurs enregistrements contigus : il suffit pour cela de cliquer sur le premier enregistrement à sélectionner puis d'appuyer sur la touche **Majuscule** avant de cliquer sur le dernier. Pour sélectionner des enregistrements non adjacents, il suffit de cliquer sur chaque enregistrement en maintenant enfoncée la touche **Ctrl** (sous Windows) ou **Commande** (sous MacOS).

Pendant et après l'exécution d'un VISUALISER SELECTION, les enregistrements sélectionnés par l'utilisateur sont conservés dans un ensemble système nommé UserSet. Après l'exécution de la commande, l'ensemble UserSet est accessible pendant un VISUALISER SELECTION aux méthodes objet de boutons, aux méthodes appelées par des commandes de menu, ainsi que pour la méthode projet qui avait appelé VISUALISER SELECTION.

Exemples

(1) L'exemple suivant sélectionne tous les enregistrements de la table [Personnes]. La commande VISUALISER SELECTION est alors utilisée pour afficher les enregistrements et permettre à l'utilisateur de désigner ceux qu'il souhaite imprimer. Enfin, les enregistrements sélectionnés sont récupérés à l'aide de la commande UTILISER ENSEMBLE et imprimés avec IMPRIMER SELECTION :

```
      TOUT SELECTIONNER([Personnes]) ` Sélection de tous les enregistrements
⇒     VISUALISER SELECTION ([Personnes]; *)` Affichage des enregistrements
         ` Utiliser uniquement les enregistrements sélectionnés par l'utilisateur
      UTILISER ENSEMBLE ("UserSet")
      IMPRIMER SELECTION ([Personnes]) ` Imprimer les enregistrements sélectionnés
```

(2) Reportez-vous à l'exemple n°6 de la commande Evenement formulaire ; il indique tous les tests que vous pourrez avoir besoin d'effectuer pour surveiller la totalité des événement intervenant pendant l'exécution de la commande VISUALISER SELECTION.

(3) Pour reproduire, par exemple, les fonctionnalités apportées par le menu **Sélection** du mode Utilisation lorsque vous utilisez MODIFIER SELECTION ou VISUALISER SELECTION en mode Menus créés, procédez de la manière suivante :

I. Dans le mode Structure, créez une barre de menus comportant les menus qui vous intéressent (par exemple Tout montrer, Recherche et Trier).
II. Associez cette barre de menus (à l'aide d'un numéro de barre de menus négatif) au formulaire sortie utilisé avec les commandes VISUALISER SELECTION ou MODIFIER SELECTION.

III. Associez les méthodes projet suivantes à vos commandes de menu :

```
    ` M_TOUT_MONTRER (associée à la commande de menu Tout montrer)
$vpCourTable:=Table du formulaire courant
TOUT SELECTIONNER($vpCourTable->)

    ` M_Recherche (associée à la commande de menu Recherche)
$vpCourTable:=Table du formulaire courant
CHERCHER($vpCourTable->)

    ` M_TRIER (associée à la commande de menu Trier)
$vpCourTable:=Table du formulaire courant
TRIER($vpCourTable->)
```

Vous pouvez aussi utiliser d'autres commandes telles que IMPRIMER SELECTION, ETAT, etc., afin de reproduire les commandes de menu "standard"à chaque fois que vous affichez ou modifiez une sélection en mode Menus créés. Grâce à la commande Table du formulaire courant, ces méthodes sont génériques et les barres de menus auxquelles elles sont associées peuvent être rattachées à tout formulaire de sortie ou à toute table.

Référence

Evenement formulaire, MODIFIER SELECTION, Présentation des ensembles.

MODIFIER SELECTION

Sélections

version 6 (Modifiée)

MODIFIER SELECTION ({table}{; *}{; *})

Paramètre	Type		Description
table	Table	→	Table à afficher et modifier ou Table par défaut si ce paramètre est omis
*		→	Utiliser formulaire sortie pour un seul enregistrement et cacher les barres de défilement dans le formulaire entrée
*		→	Afficher les barres de défilement dans le formulaire entrée (= annuler le second effet du premier paramètre *)

Description

La commande MODIFIER SELECTION est quasiment identique à la commande VISUALISER SELECTION. Reportez-vous à la commande VISUALISER SELECTION pour une description détaillée.
Les seules différences entre ces deux commandes sont les suivantes :

1. VISUALISER SELECTION vous permet d'afficher les enregistrements de la sélection courante de table dans le formulaire sortie courant, ou dans le formulaire entrée lorsque vous double-cliquez sur un enregistrement. Avec MODIFIER SELECTION, vous pouvez modifier un enregistrement dans le formulaire entrée lorsque vous double-cliquez dessus (s'il n'est pas déjà chargé par un autre utilisateur/process).

2. VISUALISER SELECTION place automatiquement la table en mode Lecture seulement. MODIFIER SELECTION place automatiquement la table en mode Lecture-écriture. Les deux commandes restaurent l'état précédent de la table lorsque leur exécution est terminée.

Référence

Evenement formulaire, Présentation des ensembles, VISUALISER SELECTION.

APPLIQUER A SELECTION ({table; }formule)

Paramètre	Type		Description
table	Table	→	Table dans laquelle appliquer la formule ou Table par défaut si ce paramètre est omis
formule	Formule	→	Ligne de code ou méthode

Description

La commande APPLIQUER A SELECTION applique formule à chaque enregistrement de la sélection courante de table. La formule peut être une ligne d'instructions ou une méthode (dans ce cas, le nom de la méthode doit être saisi sans ""). Si formule entraîne la modification d'un enregistrement de table, l'enregistrement modifié est sauvegardé. Si formule ne modifie pas d'enregistrement, aucune sauvegarde n'est réalisée. Si la sélection courante est vide, APPLIQUER A SELECTION ne fait rien. La formule peut faire appel à un champ d'une table liée si le lien est automatique.

La commande APPLIQUER A SELECTION peut être utilisée pour récupérer et traiter des informations sur une sélection d'enregistrements (par exemple, calcul d'un total), ou pour modifier une sélection (par exemple, mettre en majuscule la première lettre d'un champ). Si cette commande est utilisée à l'intérieur d'une transaction, toutes les opérations réalisées pourront être annulées si la transaction n'est pas validée.

4D Server : Le serveur n'exécute aucune des commandes passées dans formule. Chaque enregistrement de la sélection est renvoyé sur le poste client pour traitement et modification.

Un thermomètre de progression s'affiche pendant l'exécution d'un APPLIQUER A SELECTION. Un appel préalable à la commande SUPPRIMER MESSAGES permet de supprimer ce thermomètre. Lorsque le thermomètre de progression est affiché, l'utilisateur peut annuler l'opération.

Exemples

(1) L'exemple suivant met en majuscule tous les noms de la table :

```
⇒ APPLIQUER A SELECTION([Emp];[Emp]Nom:= Majusc([Emp]Nom))
```

(2) Lorsque APPLIQUER A SELECTION rencontre un enregistrement verrouillé et le modifie, celui-ci n'est pas sauvegardé. Tous les enregistrements verrouillés rencontrés par la commande sont placés dans un ensemble système appelé LockedSet. Après l'exécution d'un APPLIQUER A SELECTION, il est recommandé de tester l'ensemble LockedSet pour vérifier la présence d'enregistrements verrouillés. La boucle suivante s'exécute jusqu'à ce que tous les enregistrements aient été modifiés :

```
      Repeter ` Pour chaque enregistrement verrouillé
⇒        APPLIQUER A SELECTION ([Emp];[Emp]Nom:= Majusc([Emp]Nom))
            ` Sélection des enregistrements verrouillés uniquement
         UTILISER ENSEMBLE ("LockedSet")
            ` Jusqu'à ce qu'il n'y ait plus d'enregistrement verrouillé
      Jusque (Enregistrements dans ensemble("LockedSet")=0)
```

(3) Cet exemple utilise une méthode :

```
      TOUT SELECTIONNER([Emp])
⇒     APPLIQUER A SELECTION([Emp];Capitales)
```

Référence

Présentation des ensembles.

Variables et ensembles système

Si l'utilisateur clique sur le bouton Stop dans le thermomètre de progression, la variable système OK prend la valeur 0. Sinon, elle prend la valeur 1.

REDUIRE SELECTION ({table; }nombre)

Paramètre	Type		Description
table	Table	→	Table de laquelle réduire la sélection ou Table par défaut si ce paramètre est omis
nombre	Numérique	→	Nombre d'enregistrements à conserver

Description

La commande REDUIRE SELECTION crée une nouvelle sélection d'enregistrements pour table. La commande réduit la sélection aux nombre premiers enregistrements. REDUIRE SELECTION s'applique à la sélection courante de table pour le process courant. Le premier enregistrement de la nouvelle sélection courante devient l'enregistrement courant.

Exemple

L'exemple suivant établit des statistiques pour une compétition mondiale parmi les revendeurs dans plus de 20 pays. Pour chaque pays, les trois meilleurs revendeurs qui ont vendu plus de 50 000 FF de produits font partie des 100 meilleurs revendeurs dans le monde et sont recompensés. Avec peu de lignes de codes, cette requête complexe peut être effectuée en utilisant des recherches indexées :

```
ENSEMBLE VIDE([Revendeurs];"Gagnants")  ` Créer un ensemble vide
SCAN INDEX([Revendeurs]Montant;100;<)  ` Chercher à la fin de l'index
NOMMER ENSEMBLE([Revendeurs];"100 Meilleurs Revendeurs")
      ` Placer les enregistrements sélectionnés dans un ensemble
Boucle ($Pays;1;Enregistrements dans table([Pays]))  ` pour chaque pays
      ` Chercher les revendeurs dans ce pays
   CHERCHER([Revendeurs];[Revendeurs]Pays=NomPays;*)
      ` ...qui ont vendu pour plus de 50000 F
   CHERCHER(&;[Revendeurs];[Revendeurs]Montant vendu>=50000)
      ` Les placer dans un ensemble
   NOMMER ENSEMBLE([Revendeurs];"GagnantsRevendeurs")
      ` Ils doivent être placés dans le groupe des 100 meilleurs revendeurs
   INTERSECTION("GagnantsRevendeurs";"100 Meilleurs Revendeurs";
                                                   "GagnantsRevendeurs")
   UTILISER ENSEMBLE("GagnantsRevendeurs") ` Gagnants potentiels pour le pays
      ` Trier les résultats en ordre décroissant
   TRIER([Revendeurs];[Revendeurs]Montant vendu;<)
```

```
⇒       REDUIRE SELECTION([Revendeurs];3)  ` Garder les trois meilleurs
            ` Les gagnants pour le pays
        NOMMER ENSEMBLE([Revendeurs];"GagnantsRevendeurs")
            ` Les placer dans un ensemble des gagnants mondiaux
        REUNION("GagnantsRevendeurs";"LesGagnants";"LesGagnants")
    Fin de boucle
        ` Nous n'avons plus besoin de cet ensemble
    EFFACER ENSEMBLE("100 Meilleurs Revendeurs")
        ` Nous n'avons plus besoin de cet ensemble non plus
    EFFACER ENSEMBLE("GagnantsRevendeurs")
    UTILISER ENSEMBLE("LesGagnants")  ` Voici les gagnants
    EFFACER ENSEMBLE("LesGagnants")  ` Nous n'avons plus besoin de cet ensemble
    FORMULAIRE SORTIE([Revendeurs];"Lettre des gagnants")  ` Sélectionner la lettre
    IMPRIMER SELECTION([Revendeurs]) ` Imprimer les lettres
```

Référence

CHERCHER, Présentation des ensembles, SCAN INDEX, TRIER.

SCAN INDEX (champ; nombre{; > ou <})

Paramètre	Type		Description
champ	Champ	→	Champ indexé avec lequel "scanner" les enregistrements
nombre	Numérique	→	Nombre d'enregistrements à retourner
> ou <		→	> à partir du début de l'index < à partir de la fin de l'index

Description

La commande SCAN INDEX retourne une sélection de nombre d'enregistrements de table. Cette commande est extrêmement rapide car elle utilise l'index. Si vous passez <, SCAN INDEX retourne nombre d'enregistrements à partir de la fin de l'index (valeurs supérieures). Si vous passez >, SCAN INDEX retourne nombre d'enregistrements à partir du début de l'index (valeurs inférieures). Si vous ne passez pas le dernier paramètre, la commande retourne nombre d'enregistrements à partir du début de l'index (équivaut à passer >).

SCAN INDEX fonctionne uniquement avec des champs indexés. La commande modifie la sélection courante de la table pour le process courant, mais il n'y a pas d'enregistrement courant.

Si vous spécifiez un nombre d'enregistrements supérieur au nombre d'enregistrements présents dans la table, SCAN INDEX retourne tous les enregistrements.

Exemple

Cet exemple envoie des lettres aux 50 plus mauvais clients puis aux 50 meilleurs clients :

```
⇒ SCAN INDEX([Clients]TotalDû;50;<)  ` Obtenir la liste des 50 plus mauvais clients
  TRIER([Clients]CodePostal;>)  ` Trier par code postal
  FORMULAIRE SORTIE([Clients];"Menace")
  IMPRIMER SELECTION([Clients])  ` Imprimer les lettres
⇒ SCAN INDEX([Clients]TotalDû;50;>)  ` Obtenir la liste des 50 meilleurs clients
  TRIER([Clients]CodePostal;>)  ` Trier par code postal
  FORMULAIRE SORTIE([Clients];"Remerciement")
  IMPRIMER SELECTION([Clients])  ` Imprimer les lettres
```

Référence

CHERCHER, REDUIRE SELECTION, TRIER.

ENREGISTREMENT SELECTION {(table)}

Paramètre	Type		Description
table	Table	→	Table de laquelle réduire la sélection à un enregistrement

Description

La commande ENREGISTREMENT SELECTION réduit la sélection courante de table à l'enregistrement courant. S'il n'y a pas d'enregistrement courant, ENREGISTREMENT SELECTION ne fait rien.

Note historique : A l'origine, cette commande était utile pour "replacer" dans la sélection courante un enregistrement qui avait été empilé puis dépilé de la pile d'enregistrements pendant que la sélection de la table était modifiée. Cependant, puisque dans la version 6 de 4D, FIXER DESTINATION RECHERCHE vous permet d'effectuer une recherche sans changer la sélection ni l'enregistrement courants de la table, vous n'avez plus besoin d'empiler et de dépiler un enregistrement courant pour effectuer une recherche sur sa table. Par conséquent, ENREGISTREMENT SELECTION est moins utile, à moins que vous ne souhaitiez expressément, pour une autre raison, réduire la sélection d'une table à l'enregistrement courant.

MARQUER ENREGISTREMENTS {(nomEnsemble)}

Paramètre	Type		Description
nomEnsemble	Alpha	→	Ensemble d'enregistrements à marquer ou Ensemble Userset si ce paramètre est omis

Description

La commande MARQUER ENREGISTREMENTS permet de "surligner" des enregistrements dans un formulaire de sortie. Cette opération est identique à la sélection en mode liste, par l'utilisateur, d'enregistrement(s) à l'aide de la souris ou des raccourcis **Maj+clic** ou **Ctrl+clic** (**Commande+clic** sous MacOS).
Les enregistrements "marqués" sont mis en inverse vidéo. La sélection courante n'est pas modifiée.

Note : L'ensemble système Userset est mis à jour après le redessinement des enregistrements, c'est-à-dire après la fin de l'exécution de toute la méthode d'appel — et non immédiatement après l'exécution de la commande MARQUER ENREGISTREMENTS.

• Si vous passez un nom d'ensemble valide dans le paramètre nomEnsemble, la commande s'appliquera aux enregistrements de cet ensemble.
• Si vous omettez le paramètre nomEnsemble, la commande marquera les enregistrements de l'ensemble système Userset courant.

Exemple

Dans un formulaire en liste affiché par la commande MODIFIER SELECTION, vous souhaitez que l'utilisateur puisse effectuer des recherches, sans que la sélection courante soit modifiée. Pour cela, placez un bouton **Chercher** dans le formulaire et associez-lui la méthode suivante :

```
    FIXER DESTINATION RECHERCHE(Vers ensemble;"UserSet")
    CHERCHER
    FIXER DESTINATION RECHERCHE(Vers sélection courante)
⇒   MARQUER ENREGISTREMENTS
```

Lorsque l'utilisateur clique sur le bouton, la boîte de dialogue standard de recherche apparaît. Une fois la recherche validée, les enregistrements trouvés sont affichés en vidéo inversée, sans que la sélection courante ne soit modifiée.

48

Sélections Temporaires

Les sélections temporaires vous permettent de manipuler plusieurs sélections à la fois. Une sélection temporaire est une liste ordonnée d'enregistrements pour une table dans un process. Cette liste ordonnée d'enregistrements peut avoir un nom et est conservée en mémoire. Les sélections temporaires vous fournissent un moyen facile de garder en mémoire l'ordre et l'enregistrement courant de la sélection.

Les commandes suivantes vous permettent de travailler avec les sélections temporaires :

- COPIER SELECTION
- DEPLACER SELECTION
- UTILISER SELECTION
- EFFACER SELECTION
- CREER SELECTION SUR TABLEAU

Les sélections temporaires sont créées par les commandes COPIER SELECTION, DEPLACER SELECTION et CREER SELECTION SUR TABLEAU. Les sélections temporaires sont généralement utilisées pour travailler avec une ou plusieurs sélections, effectuer une sauvegarde puis retrouver une sélection ordonnée. Il peut y avoir plusieurs sélections temporaires pour chaque table dans un process. Pour réutiliser une sélection temporaire en tant que sélection courante, appelez UTILISER SELECTION. Lorsque vous en avez terminé avec une sélection temporaire, utilisez EFFACER SELECTION.

Les sélections temporaires peuvent avoir une portée (une aire d'action) process ou interprocess.

Une sélection temporaire est interprocess lorsque son nom est précédé des symboles (<>) — le signe “inférieur à” suivi du symbole “supérieur à”.

Note : Cette syntaxe peut être utilisée sous Windows et MacOS. Sous MacOS, vous pouvez aussi utiliser le symbole "diamant" (Option + v sur un clavier français).

La portée d'une sélection temporaire interprocess est identique à celle d'une variable interprocess. On peut accéder à une sélection temporaire interprocess à partir de n'importe quel process.

Une sélection temporaire dont le nom n'est pas préfixé par les symboles (<>) est process, c'est-à-dire qu'elle n'est disponible que dans le process où elle a été créée.

Avec 4D Client et 4D Server, une sélection temporaire interprocess n'est accessible que pour les process du client qui l'a créée. Une sélection temporaire interprocess n'est pas accessible aux autres clients.

Attention : Créer une sélection temporaire nécessite l'accès à la sélection de la table. Comme les sélections sont conservées sur le serveur et qu'un process local n'a pas accès à 4D Server, ne cherchez pas à utiliser des sélections temporaires dans un process local.

Sélections temporaires et ensembles

Voici les différences majeures entre les ensembles et les sélections temporaires :

• Une sélection temporaire est une liste ordonnée d'enregistrements, ce que n'est pas un ensemble.
• Les ensembles sont économes en mémoire car il n'ont besoin que d'un bit par enregistrement de la table. Les sélections temporaires ont besoin de 4 octets pour chaque enregistrement dans la sélection.
• A la différence des ensembles, les sélections temporaires ne peuvent pas être sauvegardées sur disque.
• Alors que les opérations standard Intersection, Reunion et Difference sont possibles pour les ensembles, les sélections temporaires ne peuvent être combinées avec d'autres sélections temporaires.

Les similitudes entre les sélections temporaires et les ensembles sont les suivantes :

• Comme un ensemble, une sélection temporaire existe en mémoire.
• Une sélection temporaire et un ensemble stockent des références aux enregistrements. Si des enregistrements sont modifiés ou détruits, la sélection temporaire ou l'ensemble peuvent n'être plus valides.
• Comme un ensemble, une sélection temporaire repère l'enregistrement courant au moment où elle est créée.

COPIER SELECTION ({table; }tempo)

Paramètre	Type		Description
table	Table	→	Table de laquelle il faut copier la sélection ou Table par défaut si ce paramètre est omis
tempo	Alpha	→	Nom de la sélection temporaire à créer

Description

COPIER SELECTION copie la sélection courante de table dans une sélection temporaire tempo. La table par défaut du process courant est utilisée si le paramètre optionnel table n'est pas spécifié. La sélection temporaire tempo contient une copie de la sélection. La sélection courante et l'enregistrement courant de table pour le process courant ne sont pas modifiés.

Une sélection temporaire ne contient pas les enregistrements, mais une liste triée des références aux enregistrements. Chaque référence à un enregistrement prend 4 octets en mémoire. Ceci signifie que lorsqu'une sélection est copiée à l'aide de la commande COPIER SELECTION, la mémoire requise est 4 octets multipliés par le nombre d'enregistrements dans la sélection. Comme les sélections temporaires restent en mémoire, il vous faut assez de mémoire pour la sélection temporaire ainsi que la sélection courante de la table pour le process.

4D Server : La sélection temporaire tempo ainsi que la sélection courante sont logées dans la mémoire du poste serveur. En conséquence, assurez-vous que le serveur dispose de suffisamment de mémoire.

Utilisez la commande EFFACER SELECTION pour libérer la mémoire utilisée par tempo.

Exemple

L'exemple suivant permet de vérifier s'il y a des factures impayées dans la table [Personnes]. La sélection est triée puis sauvegardée. Nous cherchons toutes les factures qui n'ont pas été payées. Ensuite, nous réutilisons la sélection et effaçons la sélection temporaire en mémoire :

```
TOUT SELECTIONNER([Personnes])
   `Permettre à l'utilisateur de trier la sélection
TRIER([Personnes])
   ` Stocker la sélection dans une sélection temporaire
⇒  COPIER SELECTION([Personnes];"TriéeUtilisateur")
   ` Rechercher les factures impayées
CHERCHER([Personnes];[Personnes]FactureDue=Vrai)
 ` Si un enregistrement a été trouvé
Si (Enregistrements trouves([Personnes])>0)
      ` Informer l'utilisateur
   ALERTE("Oui, quelques factures n'ont pas été réglées.")
Fin de si
   ` Réutiliser la sélection temporaire triée
UTILISER SELECTION("TriéeUtilisateur")
   ` Effacer la sélection de la mémoire
EFFACER SELECTION("TriéeUtilisateur")
```

Référence

CREER SELECTION SUR TABLEAU, DEPLACER SELECTION, EFFACER SELECTION, UTILISER SELECTION.

DEPLACER SELECTION ({table; }tempo)

Paramètre	Type		Description
table	Table	→	Table de la sélection ou Table par défaut si ce paramètre est omis
tempo	Alpha	→	Nom de la sélection temporaire à créer

Description

DEPLACER SELECTION crée la sélection temporaire tempo et y place la sélection courante de table. A la différence de COPIER SELECTION, cette commande ne copie pas la sélection, mais la déplace.

Après l'exécution de cette commande, la sélection courante de table dans le process courant est vide. En conséquence, DEPLACER SELECTION ne doit pas être utilisée lorsqu'un enregistrement est en cours de modification.

En termes d'utilisation de la mémoire, DEPLACER SELECTION est plus économique que COPIER SELECTION. En effet, COPIER SELECTION utilise 4 octets de mémoire pour chaque enregistrement de la sélection. Avec DEPLACER SELECTION, seule la référence à la sélection est déplacée.

Exemple

La méthode suivante vide la sélection courante de la table [Clients] :

```
⇒ DEPLACER SELECTION([Clients]; "AEffacer")
  EFFACER SELECTION("AEffacer")
```

Référence

COPIER SELECTION, EFFACER SELECTION, UTILISER SELECTION.

UTILISER SELECTION (tempo)

Paramètre	Type		Description
tempo	Alpha	→	Nom de la sélection temporaire à utiliser

Description

UTILISER SELECTION désigne la sélection temporaire tempo comme sélection courante pour la table à laquelle elle appartient.

Lorsque vous créez une sélection temporaire, l'enregistrement courant est aussi stocké par la sélection temporaire. UTILISER SELECTION récupère la position de cet enregistrement et en fait l'enregistrement courant. L'enregistrement courant est alors chargé. S'il a été modifié après la création de la sélection temporaire tempo, il doit être sauvegardé avant que la commande UTILISER SELECTION soit appelée, afin de ne pas perdre les informations modifiées.

• Si tempo a été créée par la commande COPIER SELECTION, la sélection temporaire est utilisée comme sélection courante de la table à laquelle elle appartient. La sélection temporaire tempo existe en mémoire jusqu'à ce qu'elle soit effacée. Pour récupérer l'espace mémoire occupé par tempo, appelez la commande EFFACER SELECTION.

• Si tempo a été créée par la commande DEPLACER SELECTION, elle est utilisée comme sélection courante de la table à laquelle elle appartient et tempo n'existe plus en mémoire.

N'oubliez pas qu'une sélection temporaire est la représentation d'une sélection courante à un instant donné. Si les enregistrements que la sélection temporaire représente sont modifiés, celle-ci devient obsolète. En conséquence, une sélection temporaire doit représenter une sélection d'enregistrements dont le contenu est relativement stable. Différents événements peuvent rendre une sélection temporaire obsolète : la modification ou la suppression d'un enregistrement appartenant à la sélection temporaire ou la modification des critères de création de la sélection temporaire.

Notez que pendant une transaction, des adresses temporaires d'enregistrements sont utilisées. Si une sélection temporaire est créée pendant une transaction, elle peut stocker des adresses qui ne seront plus valides lorsque la transaction sera validée ou annulée. En effet, dans ce cas les enregistrements ne reçoivent leur adresse finale que lorsque la transaction est validée.

EFFACER SELECTION (tempo)

Paramètre	Type		Description
tempo	Alpha	→	Nom de la sélection temporaire à effacer

Description

EFFACER SELECTION efface tempo de la mémoire et donc libère la mémoire qu'elle utilisait. EFFACER SELECTION n'affecte pas les tables, sélections courantes ou enregistrements. Comme les sélections temporaires utilisent de la mémoire, il est conseillé de les effacer si vous n'en avez plus besoin.

Si tempo a été créée par la commande DEPLACER SELECTION puis traitée à l'aide de la commande UTILISER SELECTION, elle n'existe plus en mémoire. Dans ce cas, vous n'avez pas besoin d'utiliser EFFACER SELECTION.

Référence

COPIER SELECTION, DEPLACER SELECTION, UTILISER SELECTION.

CREER SELECTION SUR TABLEAU (table; tabEnrg{; tempo})

Paramètre	Type		Description
table	Table	→	Table de la sélection
tabEnrg	Tab Entier long | Booléen	→	Tableau de n° d'enregistrements, ou Tableau de booléens (Vrai = l'enregistrement est dans la sélection, Faux = il n'y est pas)
tempo	Alpha	→	Nom de la sélection temporaire à créer, ou Appliquer la commande à la sélection courante si ce paramètre est omis ou vide

Description

La commande CREER SELECTION SUR TABLEAU construit la sélection temporaire tempo à partir :
• soit du tableau de numéros d'enregistrements absolus tabEnrg de la table table,
• soit du tableau de booléens tabEnrg ; dans ce cas, les valeurs du tableau indiquent l'appartenance (Vrai) ou non (Faux) de chaque enregistrement de table à la sélection tempo.

Si vous ne passez pas le paramètre tempo ou si vous passez une chaîne vide, la commande s'appliquera à la sélection courante de table, qui sera donc mise à jour.

Lorsque vous utilisez la commande avec un tableau d'entiers longs, tous les numéros du tableau représentent la liste des numéros d'enregistrements qui feront partie de la sélection tempo. Si un numéro est invalide (enregistrement non créé), l'erreur -10503 est générée.

Lorsque vous utilisez la commande avec un tableau de booléens, le Nième élément du tableau représente l'intégration (Vrai) ou non (Faux) de l'enregistrement numéro N dans la sélection tempo. En principe, le nombre d'éléments du tableau doit être égal au nombre d'enregistrements de table. Si le tableau est plus petit que le nombre d'enregistrements, seuls les enregistrements définis par le tableau pourront faire partie de la sélection.

Note : Avec un tableau de booléens, la commande utilise les éléments du numéro 0 au numéro N-1.

Référence

COPIER SELECTION, CREER ENSEMBLE SUR TABLEAU, EFFACER SELECTION, UTILISER SELECTION.

Gestion des erreurs

Si un numéro d'enregistrement est invalide (enregistrement non créé), l'erreur -10503 est générée. Vous pouvez intercepter cette erreur à l'aide d'une méthode installée par la commande APPELER SUR ERREUR.

49

Serveur Web

4e Dimension et 4D Server contiennent un moteur de serveur Web qui vous permet de publier des bases 4D sur le Web. Les fonctionnalités uniques et inégalées du moteur du serveur Web sont les suivantes :

• **Services Web directs**
Votre base est directement publiée sur le Web. Vous n'avez pas besoin de développer une base de données, un site Web et ensuite une interface CGI entre les deux. Votre base de données est votre site Web.

• **Sécurité des connexions**
Des options de configuration automatiques vous permettent d'accorder des autorisations d'accès spécifiques aux browsers Web ou d'utiliser le système de mots de passe intégré de 4e Dimension. Vous pouvez définir un “Utilisateur Web générique” pour simplifier la gestion des accès à l’intérieur de la base. La définition d’un dossier racine HTML par défaut vous permet de verrouiller les accès aux fichiers sur le disque.

• **Connexions SSL**
Votre serveur Web 4D peut communiquer en mode sécurisé avec les navigateurs Web, à l'aide du protocole SSL (*Secured Socket Layer*). Ce protocole, compatible avec la majorité des navigateurs Web, permet d'authentifier les intervenants et garantit la confidentialité et l'intégrité de l'information échangée.

• **Traduction en HTML en ligne et de manière transparente**
4D traduit en ligne vos formulaires et composants d'interface en pages HTML, de manière transparente et dynamique. Les pages HTML deviennent instantanément disponibles pour les browsers Web (même déjà connectés à la base). Actuellement, la plupart des systèmes de bases de données Web sont soit basés sur des CGI, soit sur des pages HTML statiques. Dans le premier cas, comme cela a été évoqué ci-dessus, vous devez effectuer trois développements : base, Web et CGI. Dans le deuxième cas, à chaque fois que vous modifiez un composant d'interface dans votre base, il vous faut lancer un utilitaire qui (re)traduit vos modifications en HTML. Dans les deux cas, les éléments Web sont créés hors ligne et nécessitent explicitement une intervention manuelle du développeur de la base et/ou du site Web. Avec 4e Dimension, vous pouvez modifier chaque élément d'interface à votre convenance. Dès que vous avez enregistré des modifications en mode Structure, elles sont répercutées dans les browsers Web de manière transparente. Lorsque vous développez et testez votre application, vous pouvez tout de suite tester le résultat dans un browser Web connecté.

• Accès dynamique aux enregistrements et aux données
4D gère les browsers Web comme des clients standard de la base 4D. Si, par exemple, vous modifiez des enregistrements dans la base locale depuis 4e Dimension ou depuis un poste client (avec 4D Server), ces enregistrements deviennent instantanément disponibles pour les browsers Web. Vous n'avez pas besoin de convertir de nouveau les enregistrements en HTML, comme c'est le cas dans les autres systèmes.

• Gestion de session et de contexte dans la base
Les browsers Web vous permettent de naviguer parmi les pages Web de façon aléatoire ; vous pouvez passer d'une page à une autre, d'un site à un autre, etc. Dans le cadre de l'utilisation de browsers Web pour exploiter une base de données en client/serveur, vous devez contrôler cette navigation afin de respecter la logique des transactions de la base. Lors de la création d'un enregistrement par exemple, la règle qui s'applique est que chaque saisie doit être validée ou annulée par l'utilisateur, et ne peut être laissée en l'état suite à la sélection d'une commande de navigation du browser. Dans ce cas en effet, la saisie resterait dans un état incertain. Le moteur de serveur Web 4D contient des fonctions intégrées de gestion des sessions et des contextes de la base. Par le biais des URLs des pages Web, ces fonctions gèrent des numéros de contexte et de sous-contexte uniques qui garantissent une totale synchronisation entre la page Web affichée dans le browser et le contexte de la connexion de la base dans 4D.

• Mode sans contexte
Dans ce mode, le serveur Web 4D est un serveur HTTP parfaitement standard : les pages Web statiques sont envoyées sans qu'il soit nécessaire de maintenir de contexte. Vous pouvez toujours accéder aux données de la base 4D et construire à la volée des pages HTML "semi-dynamiques" à l'aide de balises HTML et d'URL spéciaux, avant de les envoyer aux browsers Web.
Vous pouvez utiliser le serveur Web 4D dans le mode que vous souhaitez : mode contextuel, mode sans contexte, et passer à la volée d'un mode à l'autre en fonction de vos besoins.

• Compatiblité avec les technologies Internet avancées
Le serveur Web 4D peut envoyer des documents XML et supporte la technologie WML (*Wireless Markup Language*).

• Gestion multi-utilisateurs transparente
Les browsers Web, lorsqu'ils deviennent des clients de 4D, sont totalement considérés comme des clients de la base. Si, par exemple, vous commencez à modifier un enregistrement dans un browser Web, 4D verrouille automatiquement l'enregistrement pour qu'aucun autre client ne puisse le modifier. Lorsque vous validez ou annulez la saisie, 4D libère l'enregistrement. De plus, 4D vous permet d'effectuer la saisie des données au sein d'une transaction, comme avec 4e Dimension ou 4D Client.

• Gestion des process Web
4D dispose de plusieurs process pour gérer son architecture client/serveur sur le Web. Le process intégré principal, appelé "Server Web", gère les tentatives de connexion Web. Dès qu'un browser a obtenu l'accès à la base, sa session Web est gérée par un process séparé créé automatiquement dans ce but. Grâce à l'architecture multi-tâche intégrée de 4D, non seulement les clients Web et les clients standard peuvent effectuer simultanément des requêtes (telles que des recherches) qui sont traitées en parallèle par le moteur de la base, mais vous avez également la garantie qu'une requête adressée par un client Web particulier n'interfèrera pas avec les contextes des autres process.

• Archictecture de serveur Web optimisée
Le moteur du serveur Web 4D a les mêmes capacités que le moteur de base de données 4D. Si, par exemple, vous chargez des valeurs d'enregistrements dans des tableaux, l'opération est effectuée localement sur la machine du serveur Web, puis le résultat est envoyé en entier au client Web ayant effectué la requête. De plus, comme une connexion Web est un process 4D exécuté sur le serveur Web et disposant de toutes les fonctionnalités des process, vous pouvez lancer tous vos algorithmes 4D préférés : ils sont exécutés localement sur le serveur Web, et seul le résultat, si nécessaire, est envoyé au browser Web. Par exemple, vous pouvez construire une recherche utilisant les liens entre les tables, des ensembles, puis calculer des statistiques, et enfin retourner le résultat au browser. Comme vous pouvez compiler vos applications 4D, vous pouvez construire le plus complexe et le plus performant des systèmes de base de données Web sans pour autant devoir devenir le meilleur des experts du Web.

• Encapsulation HTML et JavaScript
Bien que 4D réalise tout ce dont vous avez besoin pour publier des bases sur le Web, vous voudrez peut-être personnaliser certains aspects de votre développement à l'aide de votre propre code HTML ou JavaScript. Par exemple, vous pouvez personnaliser la page Home de votre site/base Web avec une page HTML sophistiquée et originale. 4e Dimension vous permet d'encapsuler du code HTML et JavaScript dans votre développement 4D. Vous pouvez construire une page HTML entièrement personnalisée et l'envoyer sur le Web à l'aide de la commande ENVOYER FICHIER HTML ou ENVOYER BLOB HTML. Vous pouvez également encapsuler du HTML dans un formulaire 4D dont l'aspect final sur le browser Web sera un mélange d'objets 4D et du code HTML présents dans le formulaire. Avec le HTML encapsulé, vous pouvez implémenter du code JavaScript qui effectuera, côté browser Web, des actions et des manipulations de données sans qu'il soit nécessaire de retourner une requête au serveur.

• Lien entre les objets HTML et les objets 4D
Si vous utilisez du code HTML encapsulé dans votre développement 4D, vous devez, côté 4D, pouvoir recevoir les valeurs et données saisies dans les objets HTML. Au lieu de vous contraindre à écrire des routines complexes d'analyse du HTML, 4D vous offre un système facile et intégré pour lier des objets HTML aux variables 4D : il suffit que les objets portent le même nom. En conséquence, l'analyse et la réponse aux requêtes HTML deviennent très simple à implémenter : il suffit d'écrire du code 4D qui traite les variables 4D automatiquement remplies par les données provenant du browser Web.

• Support des CGI
Le serveur Web 4D peut très simplement utiliser des CGI, et peut également être interrogé par d'autres serveurs HTTP via des CGI.

Pour en savoir plus...
• Pour paramétrer votre machine et vos bases à publier sur le Web, reportez-vous à la section Services Web, Configuration.
• Pour apprendre comment publier une base sur le Web, reportez-vous à la section Services Web, Premiers pas (Partie I) et Services Web, Premiers pas (Partie II).
• Pour en savoir plus sur l'emploi du SSL, reportez-vous à la section Services Web, Utiliser le protocole SSL.
• Pour en savoir plus sur la sécurité des accès, reportez-vous à la section Services Web, Sécurité des connexions.
• Pour en savoir plus sur l'encapsulation HTML, reportez-vous à la section Encapsulation HTML et Javascript, ainsi qu'à la section Services Web, Balises HTML 4D.
• Pour en savoir plus sur l'interaction entre les process et le Web, reportez-vous à la section Services Web, Process de connexion Web.
• Pour en savoir plus sur le mode sans contexte, reportez-vous à la section Services Web, Mode sans contexte.
• Pour en savoir plus sur l'utilisation de CGI, reportez-vous à la section Services Web, Support des CGI.

Référence
ARRETER SERVEUR WEB, ENVOYER FICHIER HTML, FIXER LIMITES AFFICHAGE WEB, FIXER PAGE ACCUEIL, FIXER RACINE HTML, FIXER RACINE HTML, FIXER TEMPORISATION WEB, Services Web, Mode sans contexte, Services Web, Paramétrages du Serveur Web, Services Web, Sécurité des connexions, Services Web, Support des CGI, Services Web, Utiliser le protocole SSL.

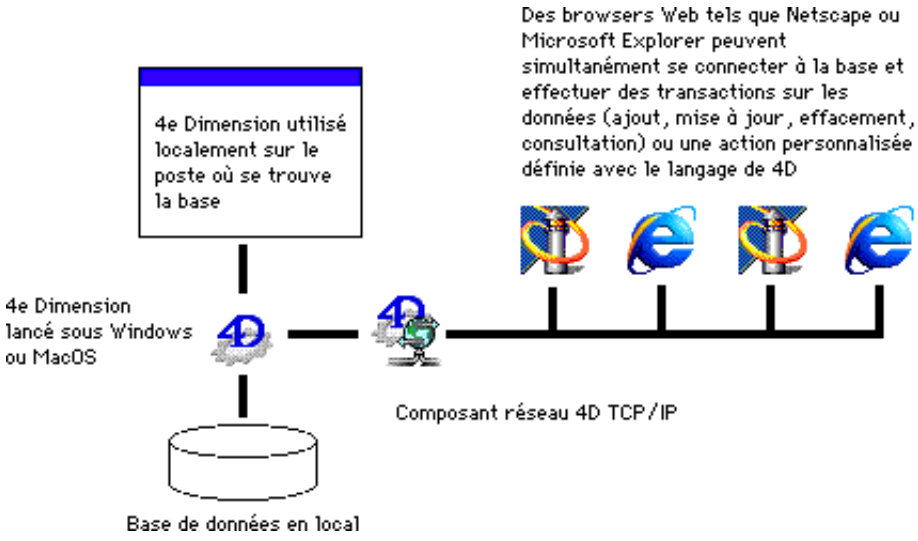
4e Dimension et 4D Server contiennent des Services Web vous permettant de publier de manière transparente et dynamique votre base sur le Web.

4e Dimension et le Web

Lorsqu'une base 4D est publiée sur le Web avec 4e Dimension, il est possible, simultanément :

- d'exploiter la base localement avec 4D
- de se connecter à la base avec un browser (navigateur) Web

Ce principe est décrit dans le schéma suivant :

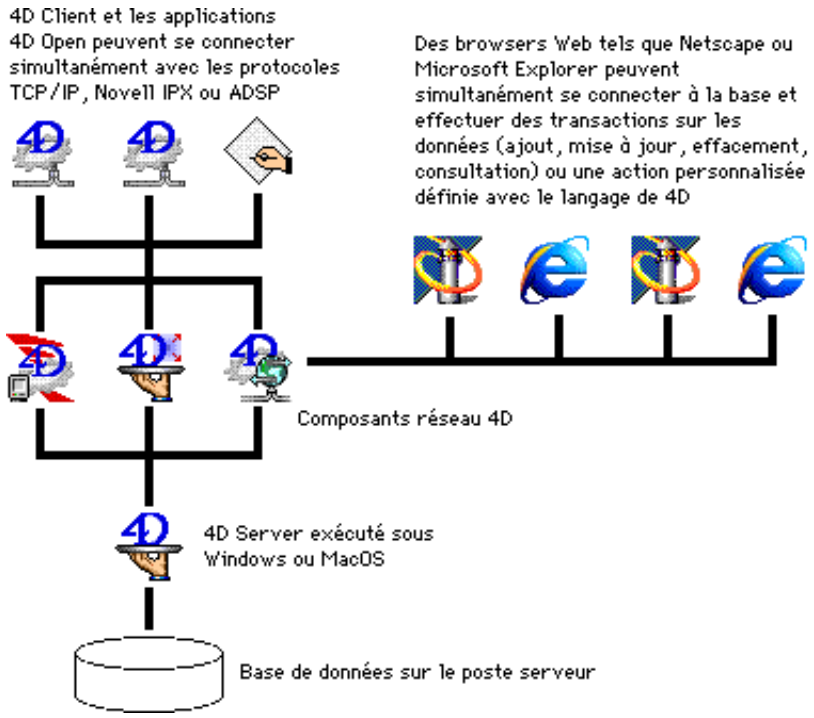


4D Server et le Web

Lorsqu'une base 4D est publiée sur le Web avec 4D Server, il est possible de se connecter à la base 4D et de l'utiliser simultanément :

- à partir de postes 4D Client
- à partir d'applications exploitant 4D Open
- à partir de browsers Web

Ce principe est décrit dans le schéma suivant :



Publier une base 4D sur le Web

Pour pouvoir publier une base 4D sur le Web à l'aide de 4e Dimension ou de 4D Server, vous devez disposer des éléments décrits ci-dessous :

• des licences 4D Web Extension ou 4D Server Web Extension, qui doivent être installées dans votre application. Pour plus d'informations sur ce point, reportez-vous au guide d'installation de 4D.

• les connexions Web sont effectuées par le biais du protocole réseau TCP/IP. Par conséquent :

- le protocole TCP/IP doit être installé et correctement configuré sur votre machine. Reportez-vous à la documentation de votre ordinateur ou de votre système d'exploitation pour plus d'informations sur ce point.

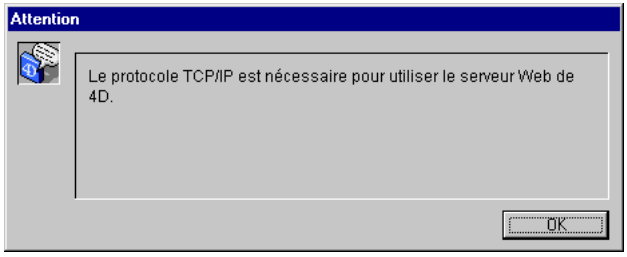
- le composant réseau 4D TCP/IP doit être installé dans votre environnement 4D : sous MacOS, le composant réseau s'installe directement dans 4e Dimension, 4D Server ou une application personnalisée fusionnée avec 4D Engine. Sous Windows, le composant réseau s'installe dans le sous-dossier 4D\NETWORK de votre dossier WINDOWS actif.

- Si vous voulez utiliser le protocole SSL pour vos connexions, assurez-vous que les composants nécessaires sont correctement installés (reportez-vous à la section Services Web, Utiliser le protocole SSL).

Note : Dans tous les cas, reportez-vous au manuel "Composants réseau pour 4D Server" pour plus d'informations.

• Une fois les deux points précédents contrôlés et réglés, vous devez démarrer les services Web depuis 4D. Ce point est traité dans le paragraphe suivant.

Si vous tentez de démarrer les services Web de 4D alors que le protocole TCP/IP ou le composant réseau 4D TCP/IP n'est pas présent, 4e Dimension affiche la boîte de dialogue d'alerte suivante :

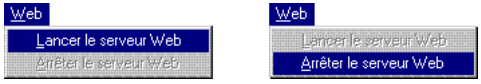


Si ce message apparaît, effectuez les installations décrites ci-dessus ou vérifiez votre configuration TCP/IP.

Démarrer les Services Web 4D

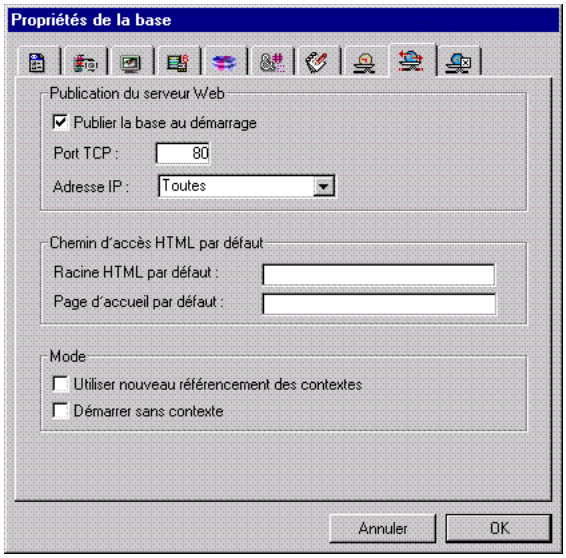
Les services Web 4D peuvent être démarrés de trois manières différentes :

• par l'intermédiaire du menu **Web**, situé dans la barre de menus de 4D Server et dans la barre de menus du mode Utilisation de 4e Dimension. Ce menu vous permet de lancer et d'arrêter le serveur Web à tout moment :



• par la publication automatique de la base à chaque fois qu'elle est ouverte. Pour que la base soit automatiquement publiée comme serveur Web, choisissez la commande **Propriétés de la base...** dans le menu **Fichier** de 4D Server ou de 4e Dimension (mode Structure).

La fenêtre des propriétés de la base s'affiche. Cliquez sur l'onglet **Serveur Web I** :



Dans la zone “Publication du serveur Web”, cochez la case **Publier la base au démarrage** puis cliquez sur le bouton **OK**. La base sera désormais automatiquement publiée comme serveur Web chaque fois que vous l'ouvrirez avec 4e Dimension ou 4D Server.

• par programmation, en appelant la commande LANCER SERVEUR WEB.

Note : Il n'est pas nécessaire de réouvrir votre base de données pour lancer ou arrêter sa publication comme serveur Web. Vous pouvez interrompre et redémarrer les services Web autant de fois que vous voulez à l'aide du menu **Web** ou en appelant les commandes LANCER SERVEUR WEB et ARRETER SERVEUR WEB.

Connexion à une base 4D publiée sur le Web

Une fois que vous avez lancé la publication d'une base 4D sur le Web, vous pouvez vous y connecter avec un browser Web. Pour cela :

• Si votre site Web dispose d'un nom d'hôte enregistré (par exemple, “Fleurs internationales”), il vous suffit d'indiquer ce nom dans la zone "Ouvrir", "Adresse" ou encore "Lieu" du browser puis d'appuyer sur la touche Entrée pour vous connecter.

• Si votre site Web ne dispose pas d'un nom enregistré, indiquez l'adresse IP de la machine de la base (par exemple 123.4.567.89) dans la zone "Ouvrir", "Adresse" ou encore "Lieu" du browser puis appuyez sur la touche Entrée.

A cet instant, soit vous êtes connecté (tout fonctionne parfaitement), soit vous pouvez rencontrer une des situations décrites ci-dessous.

Note : Si votre base est protégée par un système de contrôle d'accès, il se peut que vous ayez à saisir un nom et un mot de passe (pour plus d'informations, reportez-vous à la section Services Web, Sécurité des connexions).

(1) Vous obtenez un message du type "...le serveur n'accepte pas de connexions ou est occupé...". Dans ce cas, effectuez les contrôles suivants :
• Vérifiez que le nom du serveur ou l'adresse IP que vous avez saisi(e) est correct(e).
• Vérifiez que 4e Dimension ou 4D Server est bien lancé et que les services Web ont bien démarré.
• Vérifiez si la base de données est bien configurée pour être publiée sur un port TCP différent du port TCP Web par défaut (sur ce point, reportez-vous au paragraphe suivant).
• Vérifiez si le protocole réseau TCP/IP est correctement configuré sur la machine serveur et sur la machine du browser (les deux machines doivent se trouver sur le même réseau ou sous-réseau, ou les routeurs doivent être correctement configurés).
• Vérifiez les connexions physiques.
• Si vous ne testez pas localement votre propre site mais essayez de vous connecter à une base Web publiée sur Internet ou Intranet par quelqu'un d'autre, il se peut qu'en définitive le message décrive une situation réelle : le poste serveur peut être éteint ou occupé, dans ce cas vous pouvez tenter de vous connecter ultérieurement ou contacter l'administrateur du site Web.

(2) La connexion est établie, mais vous obtenez une page Web avec le message "Cette base de données n'a pas encore été configurée pour le Web". Cela signifie que vous êtes bien connecté à la base, mais que les services Web n'ont pas encore été lancés. A noter qu'une base doit comporter au minimum certains éléments pour pouvoir être consultable sur le Web. Pour plus d'informations sur ce point, reportez-vous à la section Services Web, Premiers pas (Partie I).

(3) La connexion est établie, mais vous n'obtenez pas la page Web que vous attendiez ! Cela peut se produire lorsque plusieurs serveurs Web sont exécutés simultanément sur la même machine. Par exemple :

• Vous avez lancé une seule base Web 4D, mais sur un système Windows NT 4.x qui exécute déjà ses propres services Web.

• Vous avez lancé plusieurs bases Web 4D sur la même machine.

Dans les cas décrits ci-dessus, il vous suffit de changer les numéros de port TCP sur lesquel vos bases 4D Web sont publiées. Pour cela, reportez-vous au paragraphe ci-dessous.

Changer le numéro de port TCP

Par défaut, 4D publie une base Web sur le port TCP standard du Web, qui est le port 80. Si ce port est déjà utilisé par un autre service Web, vous devez changer le port TCP utilisé par 4D pour votre base. Pour cela, choisissez la commande **Propriétés de la base...** dans le menu **Fichier** de 4D Server ou le menu **Fichier** du mode Structure de 4e Dimension (la fenêtre des propriétés de la base est présentée ci-dessus). Dans la zone de saisie “Port TCP”, indiquez le numéro de port TCP à utiliser pour cette base (c'est-à-dire un numéro de port TCP non utilisé par un autre service TCP/IP sur la machine).

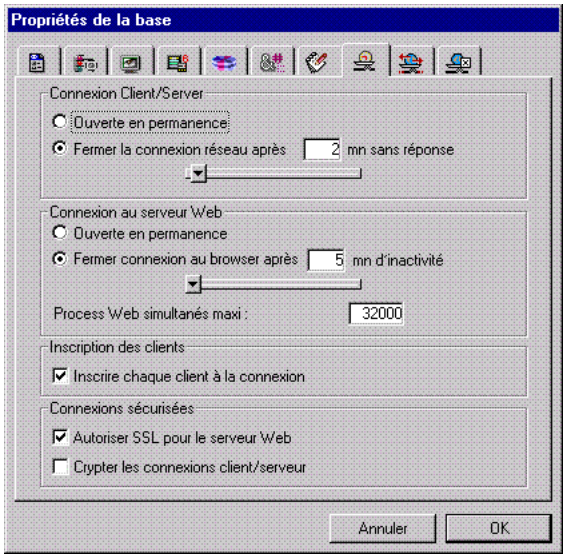
Note : Si vous passez 0, 4D utilisera le numéro de port TCP par défaut, c'est-à-dire 80.

Au niveau du browser Web, vous devez inclure ce numéro de port TCP personnalisé dans la description de l'adresse utilisée pour vous connecter à la base Web. L'adresse doit être suivie du signe “deux-points” et du numéro de port. Dans l'exemple précédent, si, par exemple, vous utiliser le port TCP numéro 8080, vous devrez spécifier dans le browser “123.4.567.89:8080”.

ATTENTION : Lorsque vous utilisez des numéros de port TCP autres que le numéro par défaut (80), prenez garde à ne pas spécifier de numéros de port qui se trouvent être les numéros par défaut d'autres services que vous employez simultanément. Si, par exemple, vous envisagez d'exploiter le protocole FTP sur la machine serveur Web, n'utilisez pas les numéros de ports TCP 20 et 21 qui sont les ports par défaut de ce protocole. Le port 443 est le port par défaut pour les connexions SSL (voir ci-desous). Pour plus d'informations sur les numéros de ports TCP par défaut et les protocoles, nous vous conseillons de vous reporter à toute documentation sur le protocole TCP/IP et d'y rechercher la table d'assignation des numéros standard RFC 1700. Les numéros de port inférieurs à 256 sont réservés pour les services standard et les numéros 256 à 1024 sont réservés pour les services spécifiques issus des plates-formes UNIX. Pour une sécurité maximum, il vous suffit de spécifier un numéro de port situé au-delà de ces intervalles, par exemple dans les 2000 ou 3000.

Activer SSL pour le serveur Web

La page “Connexions” des Propriétés de la base comporte une option permettant de définir l’utilisation du protocole de sécurisation SSL (*Secured Socket Layout*) dans le cadre du serveur Web.



L'option **Autoriser SSL pour le serveur Web** permet d’activer ou d’inactiver l’utilisation du protocole SSL pour les connexions du serveur Web. Par défaut, cette option est cochée. Le port TCP utilisé pour les connexions SSL est le port 443.
Vous pouvez désélectionner cette option si vous ne souhaitez pas exploiter les fonctionnalités SSL avec votre serveur Web, ou si un autre serveur Web autorisant les connexions sécurisées fonctionne sur le même poste.
Si vous souhaitez utiliser le protocole SSL, assurez-vous que cette option est cochée.
Pour plus d'informations sur le protocole SSL, reportez-vous à la section Services Web, Utiliser le protocole SSL.

Référence

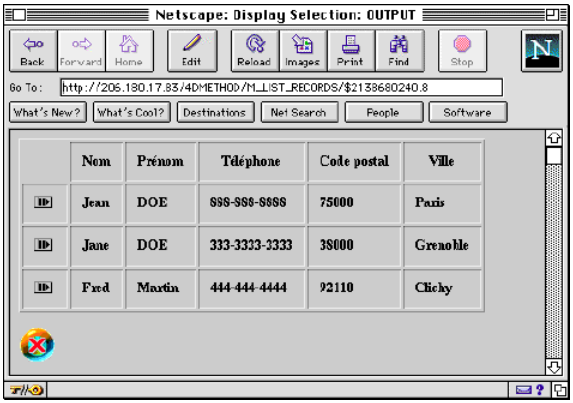
ARRETER SERVEUR WEB, ENVOYER FICHIER HTML, FIXER LIMITES AFFICHAGE WEB, FIXER RACINE HTML, FIXER TEMPORISATION WEB, Services Web, Utiliser le protocole SSL.

Exemple en mode contextuel

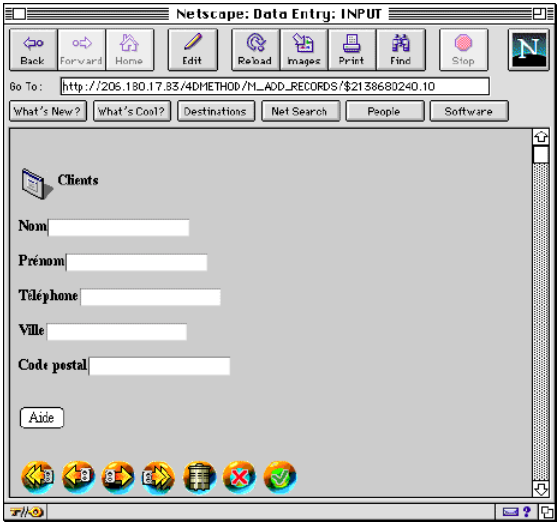
Nous allons examiner un exemple simple de base de données 4D publiée automatiquement sur le Web (en mode contextuel). Le fonctionnement et la structure de cette base sont classiques : la base comporte une table, un formulaire entrée, un formulaire sortie et une barre de menus. La page Home est personnalisée.

Lorsqu'un browser se connecte au serveur Web 4D, il obtient la page Home suivante :

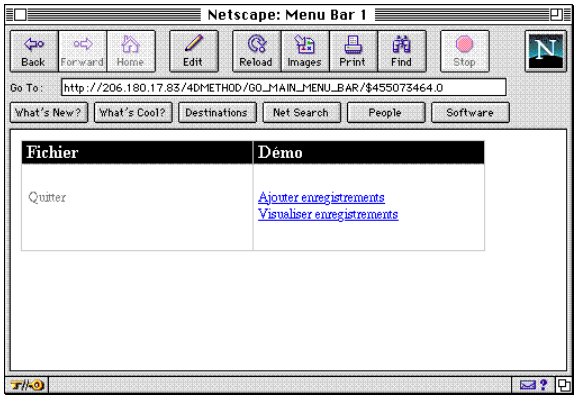
Si vous cliquez sur le lien 'Visualiser enregistrements', vous obtenez l'équivalent de l'affichage d'une sélection 4D sur le Web :



Vous pouvez dès lors naviguer à votre convenance parmi les enregistrements. Lorsque vous cliquez sur le bouton 'Terminé' (le bouton comportant un "X"), vous retournez à la page Home du site Web.
Si vous cliquez sur le lien 'Ajouter enregistrements' dans la page Home, vous obtenez l'équivalent d'un ajout d'enregistrement 4D sur le Web :



Vous pouvez ajouter autant d'enregistrements que vous voulez. Lorsque vous cliquez sur le bouton 'Terminé' (celui avec le "X") vous retournez à nouveau à la page Home du site Web.
Si vous cliquez sur le bouton **Barre de menus principale** dans la page Home, vous obtenez la barre de menus personnalisée de 4D sur le Web :

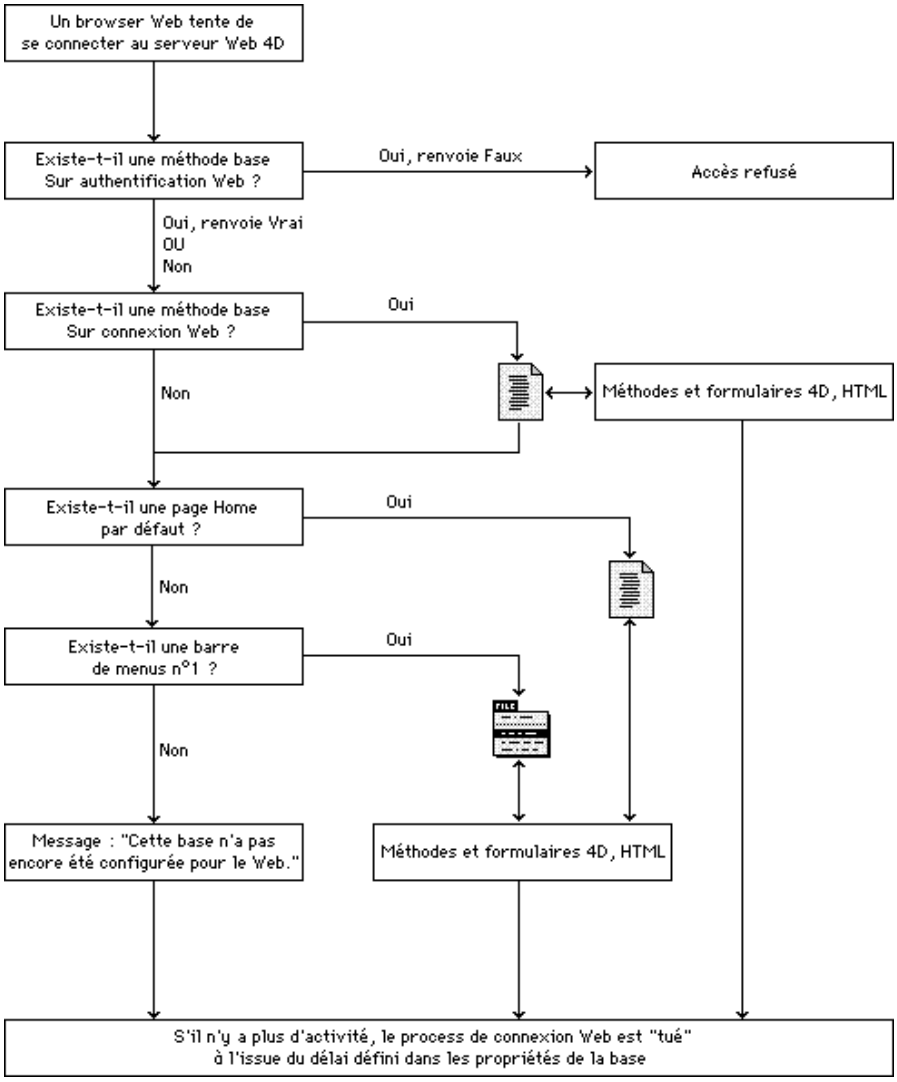


Vous pouvez alors cliquer sur une commande de menu pour visualiser ou ajouter des enregistrements. Les méthodes 4D associées à ces commandes de menu sont les mêmes que celles qui étaient utilisées à partir de la page Home.
Lorsque vous avez terminé, vous quittez le browser. 4D terminera la connexion Web dès que le délai d'attente avant déconnexion se sera écoulé.

Démarrage d'une connexion Web

Chaque fois qu'un browser Web se connecte à une base 4D publiée en tant que serveur Web, 4D effectue par défaut les actions suivantes :
• La Méthode base Sur authentification Web est exécutée, si elle existe.
• Si cette méthode base n'existe pas ou si elle retourne Vrai, la Méthode base Sur connexion Web est exécutée, si elle existe.
• Si cette méthode base n'existe pas ou si son exécution est terminée, 4D affiche la page Home par défaut définie dans les Propriétés de la base, s'il y en a une.
• Si aucune page Home par défaut n'est définie, 4D affiche la barre de menus courante (par défaut, la barre de menus n°1), si elle existe.
• S'il n'y a pas de page Home par défaut ni de barre de menus définie dans la base, 4e Dimension affiche une page de Web qui indique :*"Cette base n'a pas encore été configurée pour le Web"*.

Cette séquence est résumée dans le schéma suivant :



Note : Ce fonctionnement est celui par défaut du serveur Web 4D (**mode contextuel**). Si la base démarre en **mode sans contexte**, cette séquence automatique s'interrompt après l'appel de la page Home par défaut, si elle existe (reportez-vous à la section Services Web, Mode sans contexte).

La Méthode base Sur connexion Web peut appeler toute méthode ou tout formulaire défini(e) dans la base, ainsi que des pages HTML. En définitive, cette méthode base peut gérer la totalité de la session.

Une connexion Web à 4D ou 4D Server n'est pas de même nature qu'une connexion client/serveur. Le protocole HTTP, qui supporte HTML et le Web, n'est pas un protocole basé sur la session. C'est plutôt un protocole basé sur la requête. En client/serveur, vous vous connectez, effectuez une session de travail, et finalement vous vous déconnectez du serveur. Avec HTTP, chaque fois que vous réalisez une action qui nécessite l'attention ou une action du serveur Web, une requête est envoyée au serveur. Pour résumer, une requête HTTP peut être considérée comme une séquence connexion+requête+attente de réponse+déconnexion.

Pour maintenir une session client/serveur réelle par l'intermédiaire de HTTP, par défaut 4D gère pour vous, via un encodage transparent des URLs, des contextes qui identifient de façon unique chaque connexion Web, et en même temps associent la connexion au process 4D qui la gère : c'est le mode contextuel.

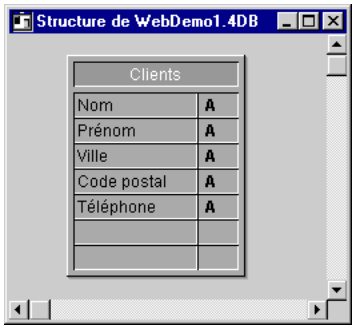
Dans ce mode toutefois, 4D ne peut pas terminer une "session de requêtes" Web de la même manière qu'une session client/serveur standard. C'est pour cette raison que la clôture de ces sessions client/serveur est gérée par un système de temporisation (*timeout*). Le process 4D qui gère la connexion Web est tué lorsque la période d'inactivité maximale autorisée pour la base est atteinte.

Base de données et serveur Web à la fois

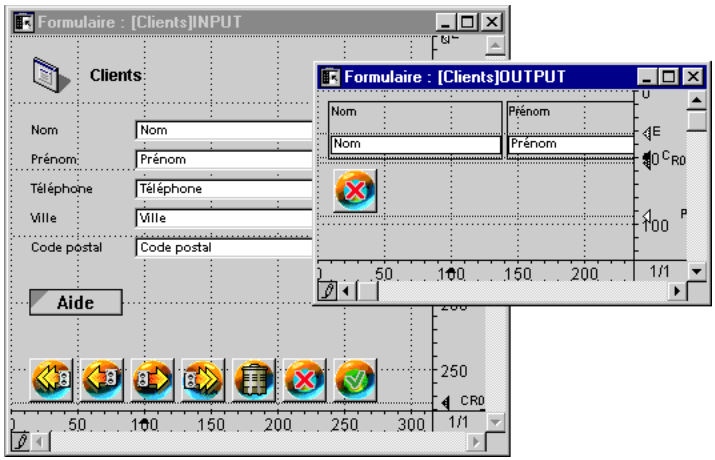
Une session de serveur Web 4D peut être entièrement gérée à l'aide des barres de menus, formulaires et méthodes 4D. Dans l'exemple précédent, la visualisation et l'ajout des enregistrements étaient effectués par l'intermédiaire de méthodes et formulaires 4D simples. Si nous n'avions pas inclus de page Home en HTML, la barre de menus n°1 aurait été directement affichée lors de la connexion Web.

Si nous mettons de côté la page HTML Home, la construction d'un serveur Web qui supporte les transactions de la base client/serveur revient à développer une base 4D comme sous Windows ou MacOS, en mono ou multi-utilisateur.

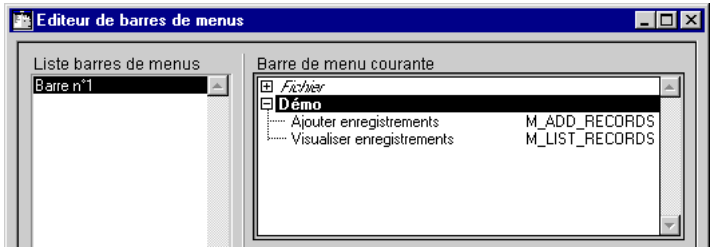
• Voici la structure de la base exemple :



• Pour visualiser les enregistrements, les formulaires entrée et sortie suivants ont été créés :



• Pour travailler avec les menus personnalisés et gérer les connexions Web, la barre de menus n°1 suivante a été créée :



• Enfin, les deux méthodes projet suivantes ont été écrites :

```
   `Méthode projet M_ADD_RECORDS
C_TEXTE ($1) `Ce paramètre doit toujours être déclaré
Repeter
   AJOUTER ENREGISTREMENT ([Clients])
Jusque (OK=0)

   `Méthode projet M_LIST_RECORDS
C_TEXTE ($1) `Ce paramètre doit toujours être déclaré
TOUT SELECTIONNER ([Clients])
MODIFIER SELECTION([Clients])
```

C'est tout !

En moins de cinq minutes, lorsque vous créez une base 4D, elle peut être utilisée localement ou comme serveur Web que vous pouvez publier sur votre réseau Intranet ou sur Internet.

Voir aussi la section Services Web, Premiers pas (Partie II).

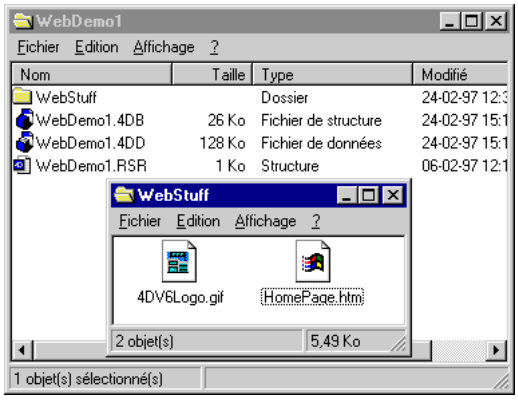
Référence

ARRETER SERVEUR WEB, ENVOYER FICHIER HTML, FIXER LIMITES AFFICHAGE WEB, FIXER LIMITES AFFICHAGE WEB, FIXER TEMPORISATION WEB, LANCER SERVEUR WEB.

Ajouter une touche de HTML à une base (en mode contextuel)

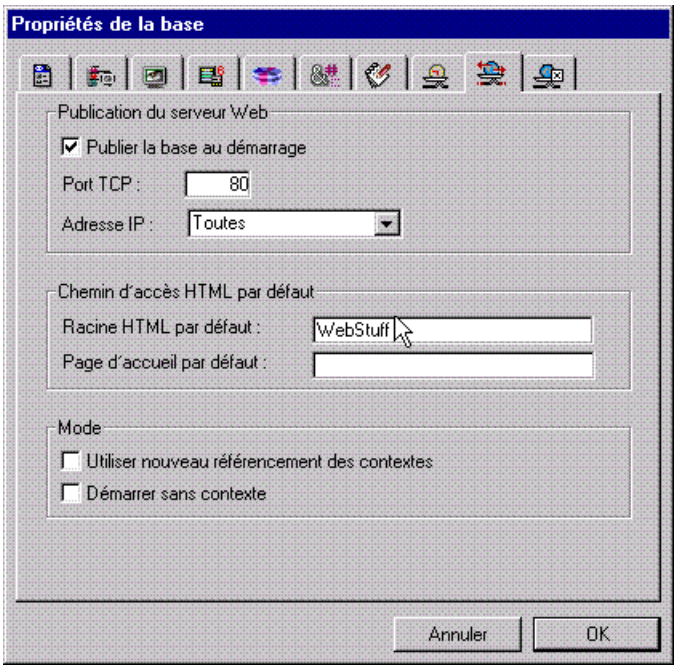
Si vous ne voulez pas que la barre de menus n°1 de votre base soit présentée aux utilisateurs Web, vous pouvez soit envoyer un formulaire 4D, soit paramétrer le serveur Web de manière à ce qu'il affiche une page d'accueil HTML par défaut. Il est intéressant de noter que vous pouvez réutiliser des pages HTML provenant d'autres sites Web que vous avez développés. Ces pages peuvent avoir été créées à l'aide de tout outil HTML.

Dans cet exemple, nous avons choisi d'utiliser une page HTML. Le répertoire de la base est le suivant :



Le document HomePage.HTM est la page HTML que nous voulons utiliser en tant que page Home pour la base. Le document 4DV6Logo.GIF est une image utilisée dans le document HTML.

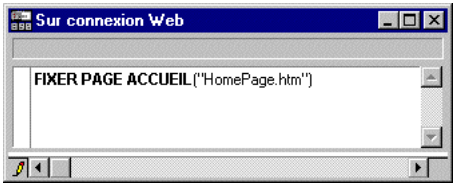
Pour paramétrer le serveur Web, il suffit d'indiquer le dossier racine HTML par défaut dans la page "Serveur Web I" de la boîte de dialogue des Propriétés de la base :



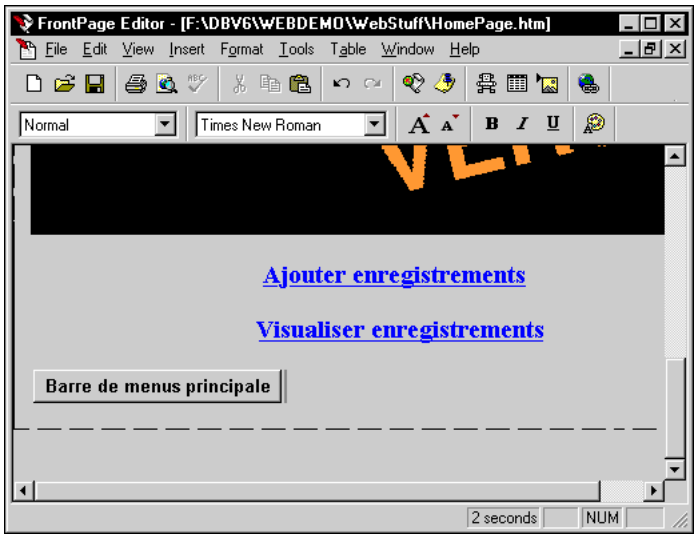
La zone **Racine HTML par défaut** indique à 4e Dimension l'emplacement par défaut des documents HTML.

Note : Pour plus d'informations sur cette boîte de dialogue, reportez-vous à la section Services Web, Paramétrages du serveur Web.

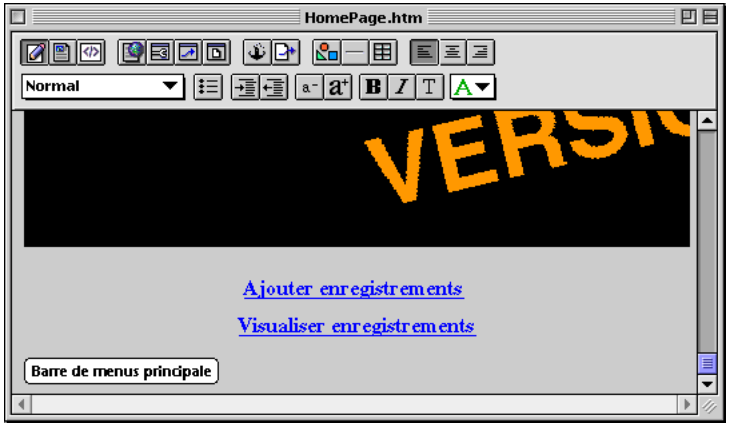
Ensuite, vous placez dans la Méthode base Sur connexion Web un appel à la commande FIXER PAGE ACCUEIL, qui vous permet de définir la page d'accueil du process Web courant :



Le document HTML utilisé dans notre exemple apparaît ainsi dans Microsoft Front Page™ :

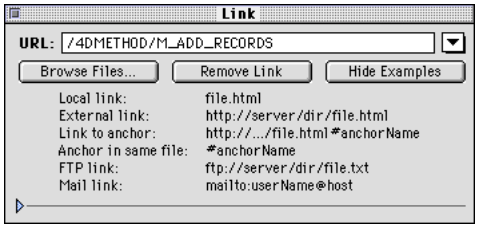


Le même document HTML apparaît ainsi dans Claris Home Page™ :

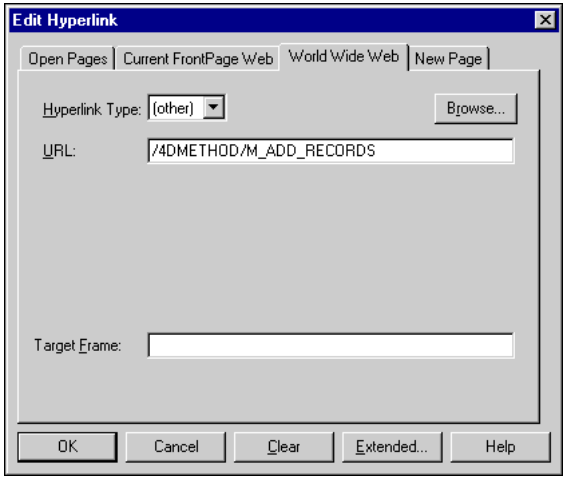


Les deux liens "Ajouter enregistrements" et "Visualiser enregistrements" déclenchent l'exécution des méthodes projet 4D M_ADD_RECORDS et M_LIST_RECORDS par l'intermédiaire de leurs URLs. La convention est simple : tout objet HTML comportant un lien peut appeler une méthode projet de votre base par l'URL "/4DMETHOD/Nom_de_votre_Méthode".

Voici l'URL du texte "Ajouter enregistrements" visualisé dans Claris Home Page™ :



Voici le même URL visualisé dans Microsoft Front Page™ :



Une fois ces liens définis, 4D exécute la méthode projet spécifiée après le mot-clé /4DMETHOD/ lorsqu'un browser Web retourne l'URL. Ensuite, lorsque la méthode projet se termine, vous retournez à la page HTML. Notez que la méthode projet peut elle-même appeler des formulaires 4D, d'autres pages HTML, et ainsi de suite. Dès qu'elle se termine, vous retournez à la page HTML qui a lancé son exécution.

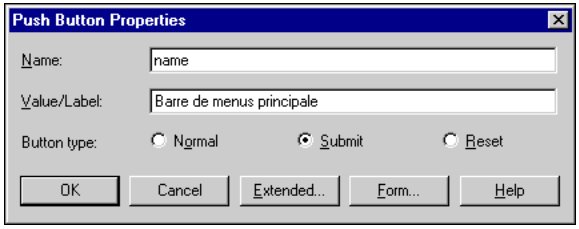
Votre site Web 4D peut être un système basé entièrement sur 4D ou un mélange de formulaires 4D et de pages HTML. Si vous utilisez des pages HTML dans votre base 4D, vous bénéficiez des deux environnements : 4D et HTML. Mais rappelez-vous que vous n'êtes absolument pas obligé d'utiliser des pages HTML, si vous le souhaitez !

Le document HTML utilisé dans l'exemple comprend également un bouton. Il existe trois types de boutons HTML : **normal**, **submit** et **reset**. Un bouton "normal" peut se voir attribuer un URL qui fait référence à une méthode 4D à l'aide du mot-clé /4DMETHOD/. Les boutons "reset" ne sont guère utiles dans un développement 4D : ils effacent les valeurs éventuellement saisies par l'utilisateur dans un formulaire et n'envoient pas de requête au serveur. Les boutons "submit" retournent le formulaire au serveur Web avec les valeurs éventuellement saisies par l'utilisateur. Lors de l'intégration des pages HTML dans 4D, vous utiliserez généralement des boutons de type "normal" ou "submit". Les boutons de type "normal" servent à naviguer parmi les pages et les boutons de type "submit" servent à gérer la saisie de valeurs à partir de pages HTML lorsque vous ne souhaitez pas utiliser pour cela de formulaires 4D standard.

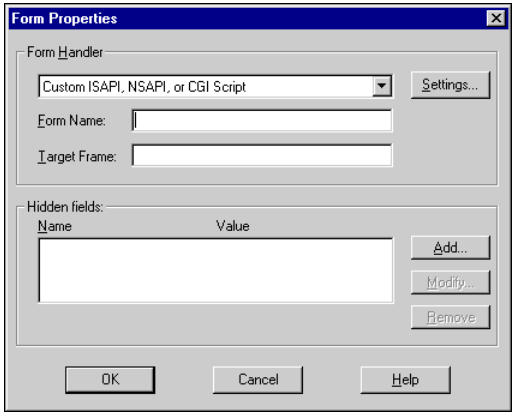
Pour gérer l'envoi du formulaire HTML du côté du serveur Web 4D, vous devez spécifier l'action POST pour la méthode 4D qui sera exécutée par 4D dès réception du formulaire.

Nous allons illustrer la définition de l'action POST pour un bouton à l'aide de l'application Microsoft Front Page™ et de Claris Home Page™.

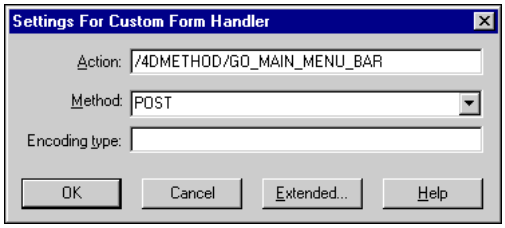
• Dans Microsoft Front Page™, demandez les propriétés du bouton "submit". La boîte de dialogue suivante s'affiche :



Cliquez sur le bouton **Form...** La boîte de dialogue des propriétés du formulaire s'affiche :

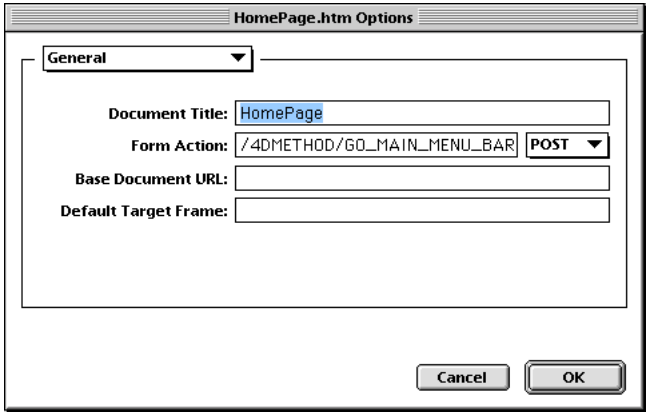


Cliquez le bouton **Settings...** pour faire apparaître la boîte de dialogue suivante :



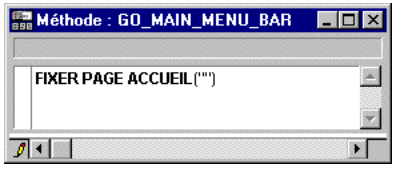
- Sélectionnez **POST** dans la liste déroulante **Method**.
- Comme pour un lien URL, spécifiez "/4DMETHOD/Nom_de_votre_Méthode" comme Action. Dans notre exemple, nous avons saisi GO_MAIN_MENU_BAR comme méthode 4D à exécuter lorsqu'un utilisateur clique sur le bouton **Barre de menus principale**.

• Dans Claris Home Page™, sélectionnez **Document Options...** dans le menu **Edition**. La boîte de dialogue suivante s'affiche :



- Sélectionnez **POST** dans le pop-up menu situé à droite de la zone Form Action,
- Saisissez "/4DMETHOD/Nom_de_votre_Méthode" comme **Form Action**.

La méthode projet GO_MAIN_MENU_BAR est la suivante :



Dans cet exemple, la méthode n'a qu'un objectif : ne plus afficher la page d'accueil par défaut, et donc envoyer la barre de menus courante. 4D affiche alors la barre de menus n°1 de votre base.

C'est tout !

En supposant que vous avez passé 5 minutes à créer la page Web et à écrire la méthode projet GO_MAIN_MENU_BAR dans votre base, vous avez obtenu en un temps record une base et un serveur Web qui combinent les capacités client/serveur et le développement HTML.

Pour en savoir plus...

• Pour plus d'informations sur les divers paramétrages du serveur Web 4D, référez-vous à la section Services Web, Paramétrages du serveur Web.

• Pour plus d'informations sur l'intégration des formulaires HTML et du code dans 4D, référez-vous à la section Encapsulation HTML et Javascript.

• Si vous éprouvez des difficultés à paramétrer votre premier serveur Web avec 4D, référez-vous au chapitre Services Web, Configuration.

Référence

ARRETER SERVEUR WEB, ENVOYER FICHIER HTML, FIXER RACINE HTML, FIXER TEMPORISATION WEB, LANCER SERVEUR WEB.

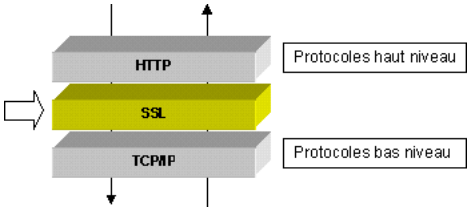
A compter de la version 6.7 de 4D, le serveur Web 4D peut communiquer en mode sécurisé via le protocole SSL (*Secured Socket Layer*).

Qu'est-ce que le protocole SSL ?

Le protocole SSL a pour but de sécuriser les communications entre deux applications — principalement un serveur Web et un browser. Ce protocole est largement répandu et compatible avec la plupart des browsers Web.

Au niveau de l'architecture réseau, le protocole SSL s'insère entre la couche TCP/IP (bas niveau) et le protocole de haut niveau HTTP, pour lequel il est principalement destiné.

Architecture réseau utilisant SSL :



Note : Le protocole SSL peut également être utilisé pour sécuriser les connexions client/serveur "classiques" de 4D Server. Pour plus d'informations, reportez-vous au manuel de référence de 4D Server.

Le protocole SSL permet de garantir l'identité de l'émetteur et du récepteur, ainsi que la confidentialité et l'intégrité des informations échangées :
• **Identification des intervenants** : l'identité de l'émetteur et du récepteur sont confirmées.
• **Confidentialité des informations échangées** : les données envoyées sont cryptées afin de les rendre inintelligibles pour les tiers non autorisés.
• **Intégrité des informations échangées** : les données reçues n'ont pas été altérées, frauduleusement ou accidentellement.

Les principes de sécurisation utilisés par SSL sont basés sur l'emploi d'un algorithme de cryptage utilisant une paire de clés : une clé privée et une clé publique.
La clé privée est utilisée pour crypter les données. Elle est conservée par l'émetteur (le site Web). La clé publique est utilisée pour décrypter les données. Elle est diffusée auprès des récepteurs (les browsers Web), via le certificat. L'emploi du SSL dans le cadre d'Internet requiert en effet l'entremise d'un opérateur de certification tel que, par exemple, Verisign®. Moyennant une participation financière du site Web demandeur, cet organisme délivre un certificat, garantissant l'identité du serveur et contenant la clé publique permettant la communication en mode sécurisé.

Note : Pour plus d'informations sur les principes généraux de cryptage et d'emploi de clés publiques/clés privées, reportez-vous à la description de la commande CRYPTER BLOB.

Installation et activation de SSL dans 4D

L'intégration du protocole SSL dans 4D a été réalisée au niveau des composants réseau. Plusieurs éléments doivent être présents sur le poste, à différents emplacements :

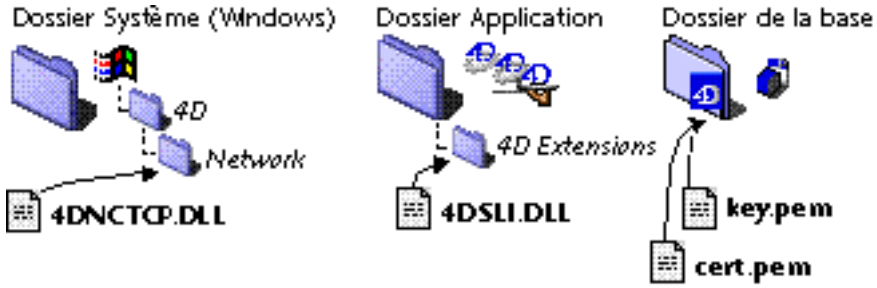
• 4DNCTCP.DLL (Windows uniquement) : DLL du composant réseau destiné au protocole TCP/IP. Sous Windows, ce fichier doit se trouver dans le dossier C:\Windows\4D\Network (où C:\Windows représente le dossier des fichiers système de Windows). Sous MacOS, les composants réseau sont intégrés aux applications 4D.

• 4DSLI.DLL : interface de la couche sécurisée (*Secured Layer Interface*) dédiée à la gestion du SSL. Ce fichier doit se trouver dans le dossier [4D Extensions] de l'application 4D qui publie la base sur le Web.

• key.pem (serveur Web uniquement) : document contenant la clé de cryptage privée. Ce fichier doit se trouver dans le dossier de la base.

• cert.pem (serveur Web uniquement) : document contenant le "certificat" (cf. paragraphe "Obtenir un certificat SSL" ci-dessous).
Ce fichier doit se trouver dans le dossier de la base.

Emplacement des fichiers nécessaires au fonctionnement de SSL avec le serveur Web 4D :



Note : La présence du composant 4DSLI.DLL est également nécessaire pour l'utilisation des commandes de cryptage des données CRYPTER BLOB et DECRYPTER BLOB.

Une fois ces éléments installés, les connexions au serveur Web ainsi que, éventuellement, les connexions client/serveur, peuvent s'effectuer en mode sécurisé.

Par défaut, les connexions SSL sont activées pour le serveur Web et inactivées pour les connexions client/serveur. Ces paramétrages sont accessibles dans la page "Connexions" des Propriétés de la base. Pour plus d'informations sur ce point, reportez-vous à la section Services Web, Paramétrages du Serveur Web.

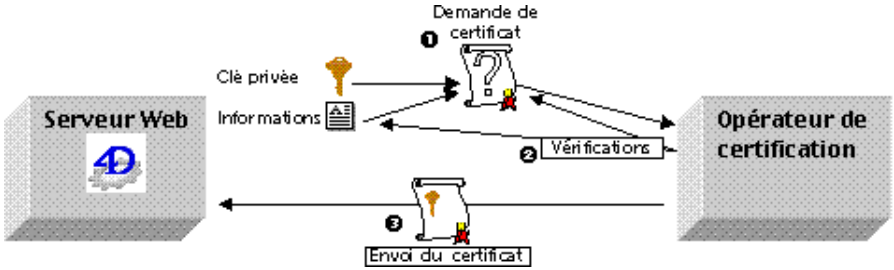
Le port TCP réservé aux communications SSL est le 443. Par conséquent, vous ne pouvez pas installer plus d'un serveur Web 4D utilisant SSL par poste. Le port TCP défini dans la page "Serveur Web I" des Propriétés de la base sera utilisé pour les connexions du serveur Web en mode standard.

De manière générale, les Propriétés de la base définies pour la gestion du serveur Web 4D (mots de passe, délai avant déconnexion, taille du cache, etc.) restent appliquées, que le serveur fonctionne en mode sécurisé ou non.

Obtenir un certificat SSL

La mise en place d'un serveur Web 4D fonctionnant en SSL nécessite un certificat numérique délivré par un opérateur de certification. Ce certificat renferme diverses informations dont la carte d'identité du site ainsi que la clé publique utilisée pour communiquer avec lui. Il est transmis aux browsers Web se connectant au site. Une fois qu'il est accepté, la communication en mode sécurisé s'établit.

Note : Pour qu'un browser accepte les certificats d'une autorité de certification, celle-ci doit être répertoriée dans ses Propriétés.



Le choix de l'autorité de certification dépend de plusieurs facteurs. Plus l'autorité est "connue", plus le nombre de browsers acceptant les certificats qu'elle délivre sera important, mais plus le prix à payer sera élevé. Verisign® est une des autorités de certifications les plus utilisées.

Pour obtenir un certificat SSL :

1. Générez une "clé privée" à l'aide de la commande GENERER CLES CRYPTAGE.

Attention : Pour des raisons de sécurité, la clé privée ne doit JAMAIS être diffusée sur un réseau. En fait, elle ne doit pas quitter le poste serveur Web. Le fichier Key.pem doit être placé dans le dossier de la structure de la base.

2. Etablissez une demande de certificat à l'aide de la commande GENERER DEMANDE CERTIFICAT.

3. Envoyez la demande de certificat à l'autorité de certification que vous avez choisie. Pour remplir la demande de certificat, il vous sera peut-être nécessaire de contacter l'autorité de certification. Les autorités de certification vérifient la réalité des informations qui leur ont été transmises.
La demande de certificat est générée dans un BLOB au format PEM (*Privacy Enhanced Mail*). Ce format autorise le copier-coller des clés sous forme de texte et leur envoi par E-mail sans risque d'altération de leur contenu. Vous pouvez donc par exemple sauvegarder le BLOB contenant la demande de certificat dans un document texte (à l'aide de BLOB VERS DOCUMENT), puis l'ouvrir et copier-coller son contenu dans un E-mail ou un formulaire Web destiné à l'autorité de certification.

4. Une fois que vous avez reçu votre certificat, créez un fichier texte que vous nommerez "Cert.pem" et copiez dans ce fichier le contenu du certificat.
Vous pouvez recevoir votre certificat sous plusieurs formes (généralement via un E-mail ou un formulaire HTML). Le serveur Web 4D accepte la plupart des formats de texte (MacOS, PC, Linux...) pour les certificats. En revanche, le certificat doit être au format PEM.

5. Placez le fichier “cert.pem” dans le dossier contenant la structure de la base.
Le serveur Web peut dès lors fonctionner en mode SSL. La durée de validité d’un certificat varie généralement entre six mois et un an.

Connexions des browsers en SSL

Lorsqu’un browser se connecte à un serveur Web fonctionnant en mode sécurisé, une boîte de dialogue d’alerte le lui signale. Une fois la boîte de dialogue validée, le certificat est envoyé au browser par le serveur Web.

Les deux parties “négocient” alors l’algorithme de cryptage qui va être utilisé pour la connexion. Le serveur Web 4D dispose de plusieurs algorithmes de cryptage symétriques (RC2, RC4, DES...). L’algorithme commun le plus puissant est utilisé.

Attention : Le niveau de cryptage autorisé est soumis à la législation en vigueur dans le pays d’utilisation. Les niveaux de cryptage proposés par le serveur Web 4D dépendent de la version de la bibliothèque système de cryptage utilisée. Par défaut, 4D SA fournit une version “Export” de la bibliothèque, dont les algorithmes sont limités à 40 bits.

Du côté du browser, deux indications permettent d’identifier la nature sécurisée de la connexion :
• L’URL affiché dans la zone d’adresse du browser débute par “HTTPS:” au lieu de “HTTP:”. Le nom de protocole “https” indique que la connexion s’effectue en mode SSL.
• Un symbole représentant un cadenas fermé s’affiche en bas de la fenêtre du navigateur. L’utilisateur peut double-cliquer sur ce cadenas pour obtenir des informations sur la connexion (certificat, etc.).

Mise à jour des serveurs Web existants

L’utilisation de SSL dans le serveur Web 4D ne nécessite pas de configuration système particulière. Toutefois, vous devez tenir compte du fait qu’un serveur Web SSL peut également fonctionner en mode non sécurisé. Le changement de mode de connexion peut s’effectuer si le browser en fait la demande (il suffit par exemple à l’utilisateur, dans la zone d’URL du browser, de remplacer “HTTPS” par “HTTP”). Il revient au développeur d’interdire ou de rediriger les requêtes effectuées en mode non sécurisé — la routine Connexion Web securisee permet de connaître le mode de connexion courant. De même, dans le cadre de la mise à jour en version 6.7 SSL d’un serveur Web 4D existant, veillez à ce que les URLs de navigation complets placés dans les pages et référençant d’autres pages du même site débutent bien par “HTTPS”, sinon ils provoqueront le basculement de la connexion en mode non sécurisé.

Référence

Connexion Web securisee, CRYPTER BLOB, DECRYPTER BLOB, GENERER CLES CRYPTAGE, GENERER DEMANDE CERTIFICAT, Services Web, Paramétrages du Serveur Web.

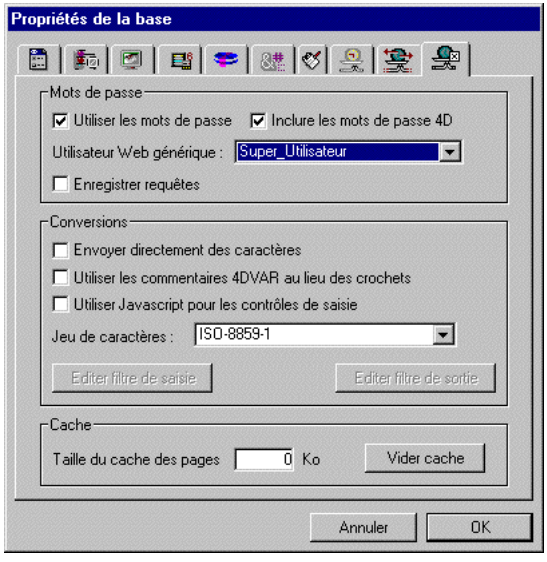
Vous pouvez sécuriser les connexions à votre serveur Web 4D à l'aide des éléments suivants :

• La combinaision du système de gestion des mots de passe pour les accès Web et de la Méthode base Sur authentification Web,
• La définition d'un "Utilisateur Web générique",
• La définition d'un dossier racine HTML par défaut.

Note : La sécurité des informations transmises via la connexion est prise en charge par le protocole SSL. Pour plus d'informations, reportez-vous à la section Services Web, Utiliser le protocole SSL.

Gestion des mots de passe pour les accès Web

Vous pouvez définir, dans les Propriétés de la base, le système de contrôle d'accès que vous souhaitez appliquer à votre serveur Web. Pour cela, dans la boîte de dialogue des Propriétés de la base, cliquez sur l'onglet "Serveur Web II". La fenêtre suivante apparaît :



Dans la zone "Mots de passe", vous disposez de deux options : **Utiliser les mots de passe** et **Inclure les mots de passe 4D**. La seconde case à cocher n'est active qui si la première a été cochée.

• **Utiliser les mots de passe** : active le système de mots de passe du serveur Web. Pour chaque connexion, une boîte de dialogue de saisie d'un nom d'utilisateur et d'un mot de passe est affichée sur le browser. Ces deux valeurs, ainsi que les paramètres de la connexion (adresse et port IP, URL...) sont envoyés à la Méthode base Sur authentification Web pour que vous puissiez les traiter.

Note : Dans ce cas, si la Méthode base Sur authentification Web n'existe pas, la connexion est rejetée.

• **Inclure les mots de passe 4D** : permet d'utiliser, au lieu ou en plus de votre propre système, le système 4D de mots de passe de la base (défini dans 4D).

Combinaison des mots de passe et de la Méthode base Sur authentification Web

Le système de filtrage des connexions au serveur Web de 4D dépend donc de la combinaison de deux paramètres :
• les options de mots de passe Web dans la boîte de dialogue des Propriétés de la base,
• l'existence ou non de la Méthode base Sur authentification Web.

Voici les différentes possibilités de contrôle des connexions :

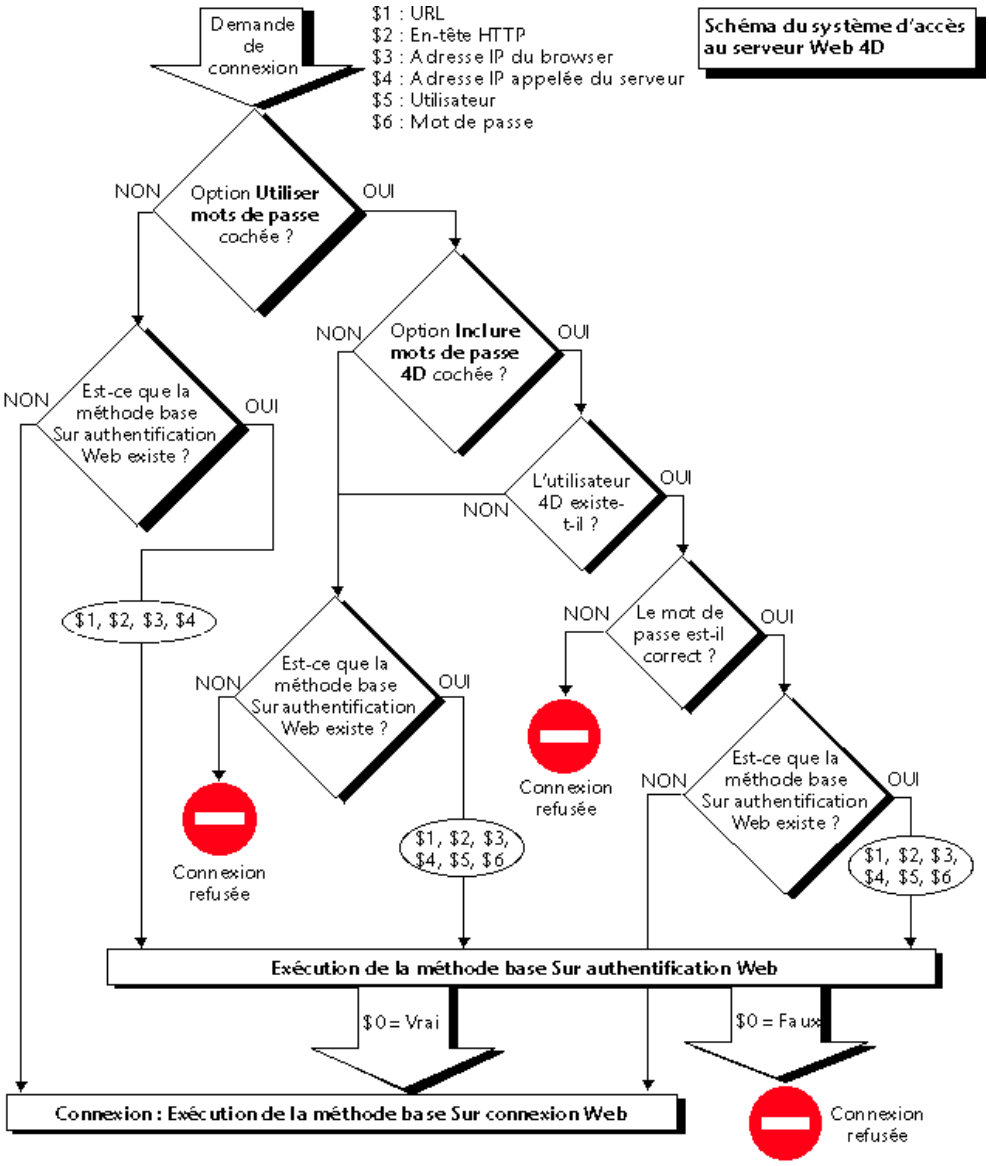
Aucune option n'est cochée
• Si la Méthode base Sur authentification Web existe, elle est exécutée et, outre $1 et $2, seules les adresses IP du browser et du serveur ($3 et $4) sont renseignées, le nom d'utilisateur et le mot de passe ($5 et $6) sont vides. Vous pouvez dans ce cas filtrer les connexions en fonction de l'adresse IP du browser et/ou de l'adresse IP demandée du serveur.
• Si la Méthode base Sur authentification Web n'existe pas, la connexion est automatiquement acceptée.

L'option "Utiliser les mots de passe" est cochée et l'option "Inclure les mots de passe de 4D" n'est pas cochée
• Si la Méthode base Sur authentification Web existe, elle est exécutée et tous ses paramètres sont renseignés. Vous pouvez dans ce cas filtrer très précisément les connexions à partir du nom d'utilisateur, du mot de passe et/ou des adresses IP du browser et du serveur Web.
• Si la Méthode base Sur authentification Web n'existe pas, la connexion est automatiquement refusée et un message indiquant que la méthode d'authentification n'existe pas est envoyé au browser.
Note : Si le nom d'utilisateur envoyé est une chaîne vide et si la Méthode base Sur authentification Web n'existe pas, une boîte de dialogue de demande de mot de passe est envoyée au browser.

Les options "Utiliser les mots de passe" et "Inclure les mots de passe de 4D" sont cochées
Note : Les bases créées avec une version de 4D antérieure à la 6.5 sont ouvertes par défaut avec ces paramètres.
• Si le nom d'utilisateur envoyé par le browser existe dans la table des utilisateurs 4D et que le mot de passe est valide, la connexion est acceptée (dans ce cas, pour des raisons de sécurité le paramètre $6 n'est alors pas renseigné). Si le mot de passe est invalide, la connexion est refusée.

• Si le nom d'utilisateur envoyé par le browser n'existe pas dans 4D, deux cas sont alors possibles :
- si la Méthode base Sur authentification Web existe, les paramètres $1, $2, $3, $4, $5 et $6 sont renseignés. Vous pouvez dans ce cas filtrer très précisément les connexions à partir du nom d'utilisateur, du mot de passe et/ou des adresses IP du browser et du serveur Web.
- si la Méthode base Sur authentification Web n'existe pas, la connexion est refusée.

Le fonctionnement du système d'accès au serveur Web 4D est résumé dans le schéma suivant :



A propos des robots (note de sécurité)

Certains robots (moteurs de recherche, spiders) parcourent les serveurs Web et les pages statiques. Si vous souhaitez que les robots ne puissent pas accéder à la totalité de votre site, il est possible de définir des URL qui leur seront interdits.
Pour cela, placez un fichier nommé ROBOTS.TXT à la racine du serveur. Ce fichier doit être structuré de la manière suivante :

User-Agent: <nom>
Disallow: <URL> ou <début d'URL>

Par exemple :

User-Agent: *
Disallow: /4D
Disallow: /%23%23
Disallow: /GIFS/

"User-Agent: *" signifie qu'il s'agit de tous les robots.
"Disallow: /4D" signifie que les robots ne doivent pas accéder aux URL commençant par /4D.
"Disallow: /%23%23" signifie que les robots ne doivent pas accéder aux URL commençant par /%23%23.
"Disallow: /GIFS/' signifie que les robots ne doivent pas accéder au dossier /GIFS/ ni aux sous-dossiers.

Autre exemple :

User-Agent: *
Disallow: /

Dans ce cas, la totalité du site est interdite aux robots.

Utilisateur Web générique

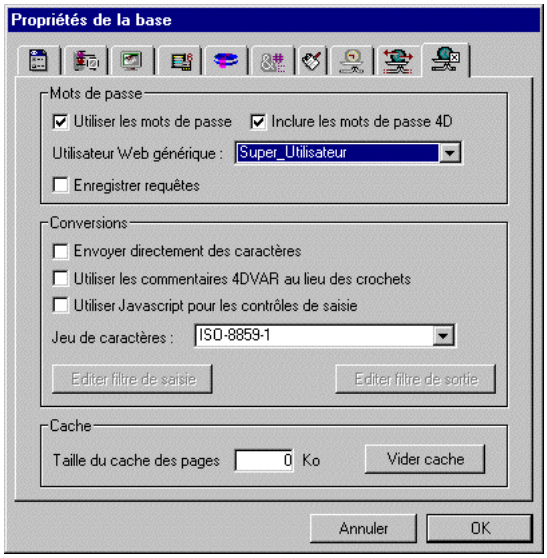
Vous pouvez désigner un utilisateur — préalablement défini dans la table des mots de passe de 4D — comme "Utilisateur Web générique". Dans ce cas, chaque browser se connectant à la base bénéficie des autorisations et restrictions d'accès associées à cet utilisateur. Vous pouvez ainsi contrôler simplement l'accès des browsers aux différentes parties de la base.

Note : Il ne faut pas confondre cette option, permettant de restreindre les accès des browsers aux différentes parties de la base (tables, menus, etc.), avec le système de contrôle des connexions au serveur Web, géré par les mots de passe et la Méthode base Sur authentification Web.

Pour définir un Utilisateur Web générique :

1. En mode Structure, créez un utilisateur dans l'Editeur de mots de passe.
Vous pouvez lui associer ou non un mot de passe.
2. Dans les différents éditeurs de 4D, assignez à cet utilisateur les autorisations et restrictions d'accès souhaitées.
3. Dans la fenêtre des Propriétés de la base, cliquez sur l'onglet "Serveur Web II".

La page suivante apparaît :



Par défaut, l'utilisateur Web générique est le Super_Utilisateur : les browsers disposent donc d'un accès libre à toutes les parties de la base.

4. Choisissez l'utilisateur dans la liste déroulante "Utilisateur Web générique" et validez la boîte de dialogue.

Tous les browsers Web autorisés à se connecter à la base bénéficieront des autorisations et restrictions d'accès associées à l'utilisateur Web générique (sauf lorsque l'option "Inclure les mots de passe 4D" est cochée et que l'utilisateur qui se connecte existe dans la table des mots de passe 4D, cf. ci-dessous).

Interaction avec le système de mots de passe Web

L'option "Utiliser les mots de passe" n'influe pas sur le mécanisme de l'utilisateur Web générique : quel que soit l'état de cette option, les privilèges et restrictions d'accès associés à l'"Utilisateur Web générique" seront appliqués à tous les browsers Web autorisés à se connecter à la base.

En revanche, lorsque l'option "Inclure les mots de passe 4D" est cochée, deux cas peuvent se produire :
• Le nom et le mot de passe de l'utilisateur n'existent pas dans la table des mots de passe de 4D. Dans ce cas, si la connexion est acceptée par la Méthode base Sur authentification Web, les droits d'accès de l'utilisateur Web générique seront appliqués au browser.
• Le nom et le mot de passe de l'utilisateur existent dans la table des mots de passe de 4D. Dans ce cas, le paramètre "Utilisateur Web générique" est ignoré : l'utilisateur se connecte avec ses propres droits d'accès.

Dossier racine HTML par défaut

Cette option des Propriétés de la base vous permet de définir le dossier dans lequel 4D recherchera les pages HTML statiques et les images à envoyer aux browsers.

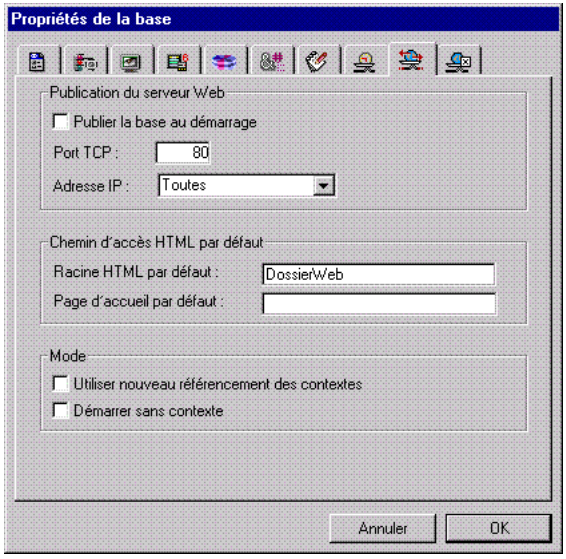
De plus, le dossier racine HTML définit, sur le disque dur du serveur Web, le niveau hiérarchique au-dessus duquel les fichiers ne seront pas accessibles. Cette restriction d'accès s'applique aux URL envoyés par les browsers Web ainsi qu'aux commandes du serveur Web 4D telles que ENVOYER FICHIER HTML. Si un URL envoyé à la base par un browser ou une commande 4D tente d'accéder à un fichier situé en amont du dossier racine HTML, une erreur est retournée, indiquant que le fichier n'a pas été trouvé.

Par défaut, 4D définit le dossier racine HTML intitulé DossierWeb, ce qui active automatiquement le système de restrictions d'accès.

Le dossier DossierWeb n'est pas créé physiquement sur le disque. Si vous conservez ce paramétrage par défaut, vous devrez, pour que les mécanismes du serveur Web 4D exploitant le dossier racine HTML par défaut fonctionnent, créer un dossier DossierWeb et le placer au niveau du fichier de structure de la base. Il vous suffira ensuite de copier les éléments requis (pages statiques, images...) dans ce dossier.

Vous pouvez également modifier le nom et l'emplacement du dossier racine HTML par défaut dans la boîte de dialogue des Propriétés de la base. Pour cela :

1. Dans la boîte de dialogue des Propriétés de la base, cliquez sur l'onglet "Serveur Web I". La page suivante apparaît :



2. Dans la zone "Racine Html par défaut", saisissez le nouveau chemin d'accès du dossier que vous souhaitez utiliser.

Le chemin d'accès saisi dans cette boîte de dialogue est relatif : il est établi à partir du dossier contenant la structure de la base. Afin d'assurer la compatibilité multi-plate-forme de vos bases, le serveur Web 4D utilise, pour décrire les chemins d'accès, des conventions d'écriture particulières. Les règles de syntaxe sont les suivantes ;
• les dossiers sont séparés par le caractère /
• le chemin ne doit pas se terminer par /
• pour "remonter" d'un niveau dans la hiérarchie des dossiers, saisissez ".." (point point) devant le nom du dossier,
• le chemin ne doit pas commencer par / (sauf si vous souhaitez que le dossier racine HTML soit le dossier de la base, cf. ci-dessous).
Par exemple, si vous souhaitez que le dossier racine HTML soit le sous-dossier "Web", placé dans le dossier "Base4D", saisissez Bases4D/Web
Si vous souhaitez que le dossier racine HTML soit le dossier de la base, mais que l'accès aux dossiers des niveaux supérieurs soit interdit, saisissez / dans la zone. Pour que l'accès aux volumes soit totalement libre, laissez la zone "Racine Html par défaut" vide.

3. Validez la boîte de dialogue.

ATTENTION : Si vous ne définissez aucun dossier racine HTML par défaut, le dossier contenant le fichier de structure de la base est utilisé. **Dans ce cas, il n'y a pas de restrictions d'accès** (tous les volumes sont accessibles).

Note : Lorsque le dossier racine HTML est modifié dans les Propriétés de la base, le cache est effacé afin de ne pas conserver des fichiers dont l'accès serait devenu restreint.

Propriétés de la base et commande FIXER RACINE HTML

Vous pouvez modifier le dossier racine HTML par défaut à l'aide de la commande FIXER RACINE HTML. Dans ce cas, la modification s'applique au process Web courant pour la session de travail. Le cache des pages HTML est alors effacé.
Toutefois, la commande FIXER RACINE HTML tient compte du dossier racine HTML par défaut. Si le dossier défini dans les Propriétés de la base est "WebPages" et si vous exécutez l'instruction FIXER RACINE HTML("Dossier"), le dossier racine HTML par défaut devient "WebPages/Dossier". Dans ce cas également, les restrictions d'accès ne sont maintenues que pour les dossiers situés en amont du dossier "WebPages".

Note : La commande FIXER RACINE HTML n'a pas d'effet lors de l'utilisation du serveur Web en mode sans contexte (cf. section Services Web, Mode sans contexte).

Référence

Méthode base Sur authentification Web, Méthode base Sur connexion Web, Services Web, Utiliser le protocole SSL.

Méthode base Sur authentification Web

Serveur Web

version 6.5

La Méthode base Sur authentification Web est chargée de gérer les accès au serveur Web. Elle est automatiquement appelée par 4e Dimension et 4D Server dès qu'un browser Web tente de se connecter à la base.

La méthode base Sur authentification Web reçoit six paramètres de type Texte, passés par 4D ($1, $2, $3, $4, $5 et $6), et retourne un booléen, $0. Voici la description des paramètres Texte :

Paramètres	Type	Description
$1	Texte	URL
$2	Texte	En-tête + Corps HTTP (dans la limite des 32 ko)
$3	Texte	Adresse IP du browser
$4	Texte	Adresse IP appelée du serveur
$5	Texte	Nom d'utilisateur
$6	Texte	Mot de passe

Vous devez déclarer ces paramètres de la manière suivante :

```
` Méthode base Sur authentification Web

C_TEXTE($1;$2;$3;$4;$5;$6)
C_BOOLEEN($0)

` Code pour la méthode
```

Note : Tous les paramètres de la Méthode base Sur authentification Web ne sont pas forcément remplis. Les informations reçues par la méthode base dépendent des options que vous avez sélectionnées dans la boîte de dialogue des Propriétés de la base. Référez-vous à la section Services Web, Sécurité des connexions.

• URL

Le premier paramètre ($1) est l'URL saisi par l'utilisateur dans la zone 'Location' de son browser Web, moins l'adresse hôte.

Prenons l'exemple d'une connexion Intranet. Supposons que l'adresse IP de votre machine serveur Web 4D est 123.4.567.89. Le tableau suivant liste les valeurs de $1 selon l'URL saisi dans le browser Web :

URL saisi dans la zone Location du browser Web	Valeur du paramètre $1
123.4.567.89	/
http://123.4.567.89	/
123.4.567.89/Clients	/Clients
http://123.4.567.89/Clients	/Clients

http://123.4.567.89/Clients/Ajouter	/Clients/Ajouter
123.4.567.89/Faire_ceci/Si_OK/Faire_cela	/Faire_ceci/Si_OK/Faire_cela

Note : Pour plus d'informations sur ce paramètre, reportez-vous à la description de la Méthode base Sur connexion Web.

• En-tête et corps de la requête HTTP
Le deuxième paramètre ($2) est l'en-tête et le corps de la requête HTTP envoyée par le browser Web. Notez que ces informations sont passées telles quelles à la Méthode base Sur authentification Web. Le contenu varie en fonction du type de browser Web qui tente de se connecter.
Si votre application exploite ces informations, il est de votre ressort d'analyser l'en-tête et le corps.

Note : Pour plus d'informations sur ce paramètre, reportez-vous à la description de la Méthode base Sur connexion Web.

• Adresse IP du browser
Le troisième paramètre ($3) reçoit l'adresse IP de la machine du browser. Cette information peut vous permettre, en particulier, de distinguer les connexions Intranet des connexions Internet.

• Adresse IP demandée du serveur
Le quatrième paramètre ($4) reçoit l'adresse IP demandée du serveur Web 4D. En effet, 4D autorise le multi-homing, permettant d'exploiter des machines disposant de plusieurs adresses IP. Pour plus d'informations sur ce point, reportez-vous à la section Services Web, Paramétrages du serveur Web.

• Nom d'utilisateur et Mot de passe
Les paramètres $5 et $6 reçoivent le nom d'utilisateur et le mot de passe saisis par l'utilisateur dans la boîte de dialogue standard d'identification affichée par le browser, le cas échéant.
Cette boîte de dialogue apparaît pour chaque connexion dès que l'option **Utiliser les mots de passe** est cochée, dans les Propriétés de la base (cf. section Services Web, Sécurité des connexions).

Note : Si le nom d'utilisateur envoyé par le browser existe dans 4D, pour des raisons de confidentialité le paramètre $6 n'est alors pas rempli (il reçoit une chaîne vide).

La Méthode base Sur authentification Web retourne un booléen dans $0 :
• Si $0 est Vrai, la connexion est acceptée.
• Si $0 est Faux, la connexion est refusée.

La Méthode base Sur connexion Web n'est exécutée que si la connexion est acceptée par Sur authentification Web.

ATTENTION : Si aucune valeur n'est passée dans $0, ou si $0 n'est pas définie dans la LIRE VARIABLES, la connexion sera considérée comme **acceptée**, et la Méthode base Sur connexion Web sera exécutée.

N'appelez aucun élément d'interface dans la Méthode base Sur authentification Web (ALERTE, DIALOGUE, etc.), sinon son exécution sera interrompue et la connexion refusée. Il en est de même si une erreur se produit durant son traitement.

Le système de filtrage des connexions au serveur Web de 4D dépend de la combinaison des options de mots de passe Web dans la boîte de dialogue des Propriétés de la base et de la Méthode base Sur authentification Web (pour plus d'informations, reportez-vous à la section Services Web, Sécurité de connexions).

Exemple

Cette Méthode base Sur authentification Web filtre les connexions à l'aide d'une table de noms d'utilisateurs et de mots de passe :

```
   `Méthode base Sur authentification Web
C_TEXTE($5;$6;$3;$4)
C_TEXTE($utilisateur;$motPasse;$IPBrowser;$IPServer)
C_BOOLEEN($utilisateur4D)
TABLEAU TEXTE($utilisateurs;0)
TABLEAU TEXTE($nums;0)
C_ENTIER LONG($upos)
C_BOOLEEN($0)

$0:=Faux

$utilisateur:=$5
$motPasse:=$6
$IPBrowser:=$3
$IPServer:=$4

   `Pour des raisons de sécurité, refuser les noms qui contiennent @
Si (AvecJoker($utilisateur) | AvecJoker($motPasse))
   $0:=Faux
      `La méthode AvecJoker est décrite ci-dessous
Sinon
      `Vérifier si c'est un utilisateur 4D
   LIRE LISTE UTILISATEURS($utilisateurs;$nums)
   $upos:=Chercher dans tableau($utilisateurs;$utilisateur)
   Si ($upos > 0)
      $utilisateur4D:=Non(Utilisateur supprime($nums{$upos}))
   Sinon
      $utilisateur4D:=Faux
   Fin de si

   Si (Non($utilisateur4D))
         `Ce n'est pas un utilisateur défini dans 4D, chercher dans la
         `table des utilisateurs Web
      CHERCHER([WebUsers];[WebUsers]User=$utilisateur;*)
      CHERCHER([WebUsers]; & [WebUsers]Password=$motPasse)
```

```
        $0:=(Enregistrements trouves([WebUsers]) = 1)
    Sinon
        $0:=Vrai
    Fin de si
Fin de si
    `Est-ce une connexion intranet ?
Si (Sous chaine($IPBrowser;1;7) # "192.100.")
    $0:=Faux
Fin de si
```

La Méthode projet *AvecJoker* est décrite ci-dessous :

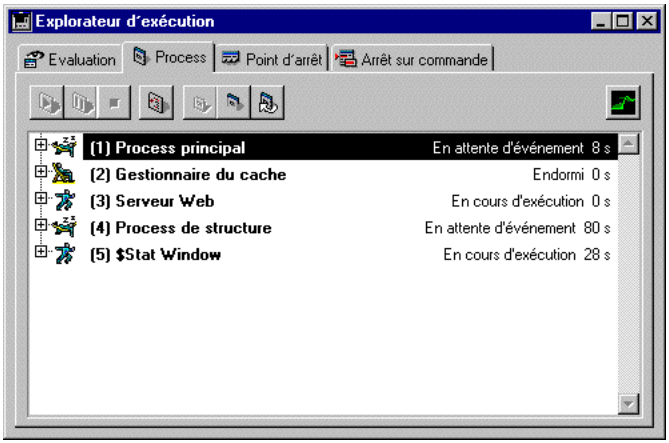
```
    `Méthode projet AvecJoker
    `AvecJoker ( Chaine ) -> Booléen
    `AvecJoker ( Nom ) -> Contient un joker

C_ENTIER($i)
C_BOOLEEN($0)
C_TEXTE($1)
$0:=Faux
Boucle($i;1;Longueur($1))
    Si (Code ascii(Sous chaine($1;$i;1)) = Code ascii("@"))
        $0:=Vrai
    Fin de si
Fin de boucle
```

Référence

Méthode base Sur connexion Web, Services Web, URLs et actions de formulaires, Services Web, Sécurité des connexions.

Process Serveur Web

Le process Serveur Web s'exécute lorsque la base est publiée en tant que serveur Web. Dans la page **Process** de l'Explorateur d'exécution présentée ci-dessous, le process Server Web est le troisième process :



Ce process est un process du noyau de 4D, vous ne pouvez donc pas l'arrêter à l'aide de la commande **Tuer**. De même, vous ne pouvez pas effectuer de communication interprocess à l'aide des commandes comme APPELER PROCESS. Notez que le process Serveur Web n'a pas d'éléments d'interface (fenêtres, menus, etc.).

Vous pouvez démarrer le process Server Web :
• en choisissant **Lancer le serveur Web** dans le menu **Web** de 4D Server ou de 4e Dimension (mode Utilisation).
• en appelant la commande 4D LANCER SERVEUR WEB.
• en ouvrant une base pour laquelle la propriété **Publier la base au démarrage** a été sélectionnée.

Vous pouvez arrêter le process Server Web :
• En choisissant **Arrêter le serveur Web** dans le menu **Web** de 4D Server ou de 4e Dimension (mode Utilisation).
• en appelant la commande 4D ARRETER SERVEUR WEB.
• en quittant une base publiée comme serveur Web.

Le rôle du process Serveur Web est de gérer les tentatives de connexion Web. Lorsque vous démarrez le process Serveur Web, vous n'ouvrez pas de connexion Web, vous permettez aux utilisateurs Web de se connecter à la base. Lorsque vous arrêtez le process Serveur Web, vous ne fermez pas les process de connexion Web ouverts (s'il y en a), simplement vous ne permettez plus à des utilisateurs Web de se connecter à la base.

Si des process de connexion Web étaient ouverts au moment où vous arrêtez le process Serveur Web, chacun de ces process continue à s'exécuter normalement, tant que l'utilisateur Web continue d'envoyer des requêtes à la base sans dépasser la période d'inactivité maximale spécifiée dans l'option “Fermer la connexion au browser” (paramètre spécifié dans la fenêtre des Propriétés de la base ou fixé par programmation à l'aide de la commande FIXER TEMPORISATION WEB).
Par conséquent, un délai d'attente peut être nécessaire avant l'arrêt complet du process Serveur Web.

Process de connexion Web

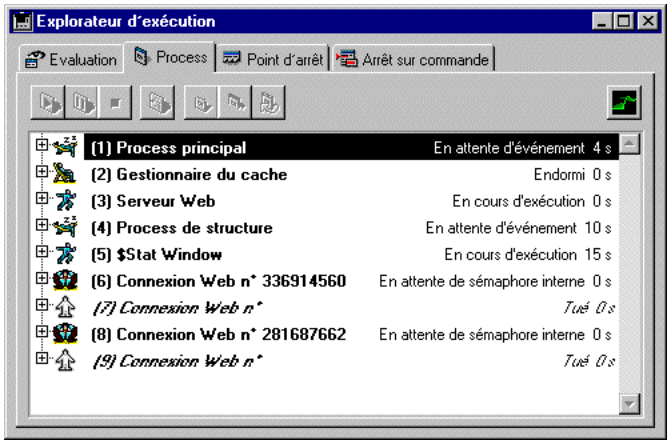
Chaque fois qu'un browser Web tente de se connecter à la base, la requête est gérée par le process Serveur Web, qui procède de la manière suivante :

• D'abord, il crée un ou plusieurs process locaux 4D temporaires pour gérer et évaluer la connexion au browser Web.

Note : Ces process temporaires gèrent toutes les requêtes HTTP. Ils s'exécutent très rapidement puis sont “tués” ou endormis. En effet, à des fins d'optimisation du serveur Web, une fois qu'il a traité une requête, un process de connexion Web temporaire est endormi pendant quelques secondes pour être éventuellement réactivé lorsqu'une autre requête arrive. Ce mécanisme peut être ajusté (délai d'attente, nombre minimum et maximum de process à conserver dans la "réserve" de process) à l'aide de la commande FIXER PARAMETRE BASE (sélecteurs 6 et 7).

• Si la requête nécessite la création d'un contexte (par exemple si la base démarre en mode contextuel), il vérifie alors si les ressources disponibles sont suffisantes. Si cela n'est pas le cas, il envoie au browser Web le message suivant : “Cette base de données n'a pas encore été paramétrée pour le Web”.

Si la connexion Web est correctement effectuée, un process **Connexion Web** est créé. Ce process gèrera toute la session Web pour cette connexion. La liste des process, présentée ci-dessous, affiche les process de connexion Web “Connexion Web n° 336914560” et “Connexion Web n° 281687662” qui sont démarrés à la suite de la connexion du browser Web :



Notez que les septième et neuvième process, qui ont été démarrés puis tués, ont géré l'initialisation des connexions Web.

Note : Pour plus d'informations sur la gestion des contextes, reportez-vous ci-dessous au paragraphe "Gestion des contextes de la connexion Web : le mode contextuel".

• Si la requête ne nécessite pas la création d'un contexte (si la base démarre en mode sans contexte), aucun contexte n'est créé, la requête est traitée et une réponse est éventuellement retournée au browser. Le process temporaire est alors tué ou endormi (cf. ci-dessus).

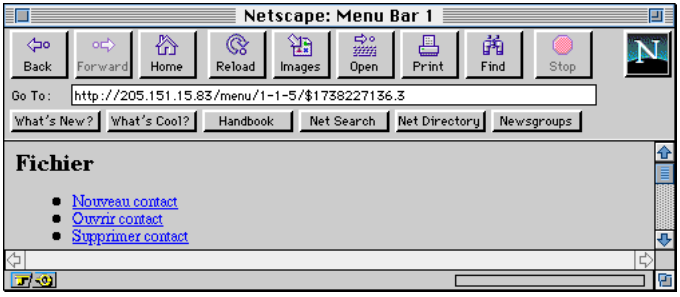
• Si au cours de la session, la connexion passe du mode contextuel au mode sans contexte, le process de connexion Web (numéroté) est immédiatement tué. A l'inverse, si au cours de la session, la connexion passe du mode sans contexte au mode contextuel, un process de connexion Web numéroté est créé.

Note : Pour plus d'informations sur le mode sans contexte, reportez-vous à la section Services Web, Mode sans contexte).

Gestion des contextes de la connexion Web : le mode contextuel

Numéro de contexte

Le numéro du process de la connexion Web s'appelle le **numéro de contexte**. C'est un nombre, généré aléatoirement, qui identifie chaque connexion Web. Le numéro de contexte est géré par 4D et par le browser pendant toute la connexion Web. Dans l'exemple ci-dessous, le numéro de contexte est 1738227136. Dans la fenêtre de votre browser Web, le numéro de contexte apparaît dans l'URL affiché dans la zone de destination :



Les URLs sont automatiquement gérés par 4D pendant toute la session Web en mode contextuel. Chaque fois qu'une requête HTTP est reçue par le composant réseau TCP/IP de 4D, 4D extrait le numéro de contexte de l'URL, et peut donc réacheminer la requête au bon process de connexion Web.

Les numéros de contexte permettent à 4D :

• de gérer les sessions Web et celles de la base de données,
• de gérer de façon transparente plusieurs connexions Web simultanées,
• d'éviter des connexions indésirables futures utilisant des marqueurs, car un numéro de contexte différent est généré à chaque connexion.

Synchronisation des sessions Web et de la base en mode contextuel : numéro de sous-contexte de la connexion Web

Dans la fenêtre précédente, vous pouvez constater que le numéro de contexte est suivi d'un point et d'un autre chiffre. Ce second chiffre s'appelle le **numéro de sous-contexte**. 4D gère et incrémente automatiquement ce numéro chaque fois qu'une nouvelle page HTML issue de 4D est envoyée au browser en mode contextuel. Le numéro de sous-contexte est essentiel à la gestion de la session de la base.

Généralement, un browser Web dispose de contrôles de navigation comme des boutons du type 'Page précédente' (Back) ou 'Page suivante' (Forward), des fenêtres d'historique, etc. Ces contrôles sont utiles lorsque vous consultez des documents, des notes d'information, des listes, etc. Ils sont moins efficaces lorsque vous effectuez une transaction à partir de la base.

Par exemple, si un utilisateur Web ajoute un enregistrement à une table, il faut savoir si la saisie des données a été validée, c'est-à-dire si l'utilisateur Web a cliqué sur le bouton **Annuler** ou **Valider** dans le formulaire 4D. Si, à ce stade, l'utilisateur Web appelle d'autres pages HTML, la saisie reste dans un état incertain. Pour cela, 4D utilise le numéro de sous-contexte afin de synchroniser la session Web côté 4D.

Lorsqu'un formulaire est soumis au browser ou qu'une requête HTTP est envoyée à 4D par le browser, si une désynchronisation des sessions du Web et de la base est détectée, 4D envoie le message “En utilisant les contrôles de navigation du browser, vous n'avez pas validé les données d'un formulaire. 4e Dimension va revenir sur ce formulaire afin que vous le validiez ou l'annuliez”. 4D retourne alors à la page Web de saisie à l'aide du numéro de sous-contexte.

La synchronisation est également essentielle pour le process de connexion Web. L'exécution d'une commande telle que, par exemple, AJOUTER ENREGISTREMENT ([...]) doit s'achever correctement pour que l'on puisse continuer à utiliser la base.

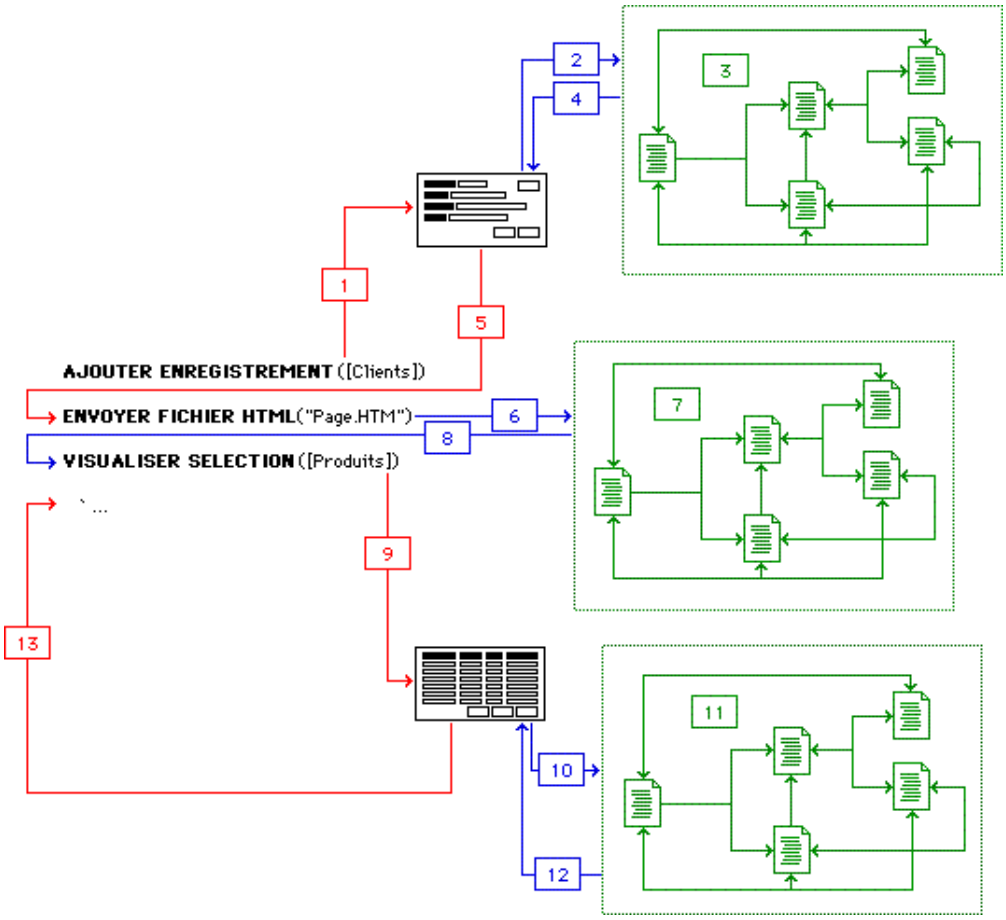
La synchronisation est sélective. Si la page Web courante affichée dans le browser est un formulaire 4D (AJOUTER ENREGISTREMENT, VISUALISER SELECTION, DIALOGUE, etc.), les contrôles de synchronisation se manifesteront le cas échéant.
Si la page Web courante est une page HTML statique issue d'un lien présent dans une autre page Web (et envoyée à l'aide de la commande ENVOYER FICHIER HTML), vous pouvez naviguer librement parmi ces pages.

Considérons les lignes de code 4D suivantes :

```
AJOUTER ENREGISTREMENT ([Clients])
ENVOYER FICHIER HTML ("Page.HTM")
VISUALISER SELECTION ([Produits])
```

Le schéma suivant décrit ce qui se passe sur 4D et sur le browser Web lors de l'exécution de ce code. Si vous consultez cette documentation sous forme électronique, notez que :

• les lignes rouges signalent les différentes transmissions du formulaire 4D.
• les lignes bleues signalent le transfert entre les pages HTML basées sur 4D et celles qui ne le sont pas.
• les lignes vertes indiquent des pages HTML qui ne sont pas basées sur 4D.



Description des étapes

(1) La commande AJOUTER ENREGISTREMENT est appelée. 4D traduit le formulaire entrée courant de la table en une page HTML et l'envoie au browser Web. Si le formulaire contient plusieurs pages, les boutons de navigation standard de 4D permettent de naviguer parmi les pages du formulaire. La navigation est implémentée et effectuée par 4D de façon transparente (via la soumission de formulaire Web).

(2) Pendant la saisie des données (donc dans l'appel à AJOUTER ENREGISTREMENT), l'utilisateur Web clique sur un bouton. La méthode objet du bouton appelle la commande ENVOYER FICHIER HTML.

(3) A la suite de l'appel à ENVOYER FICHIER HTML, si la page HTML contient des liens, l'utilisateur peut naviguer parmi différentes pages. Finalement, lorsque l'instruction ENVOYER FICHIER HTML("") est exécutée, le mode HTML est terminé.

(4) L'exécution de la méthode objet du bouton sur lequel l'utilisateur a cliqué et de la saisie des données liée à AJOUTER ENREGISTREMENT reprennent. Notez que les étapes (2) et (3) peuvent être répétées plusieurs fois pendant la saisie des données.

(5) Enfin, la saisie des données est soit validée soit annulée et le process de connexion Web est exécuté.

(6) L'appel suivant est ENVOYER FICHIER HTML.

(7) Cet étape est semblable à l'étape 3. Si la page HTML contient des liens, il est possible de naviguer parmi plusieurs pages. Lorsque finalement la commande ENVOYER FICHIER HTML("") est appelée, le mode HTML est terminé.

(8) Le process de connexion Web est exécuté.

(9) La commande VISUALISER SELECTION est appelée. 4D traduit le formulaire sortie courant de la table en page HTML et l'envoie au browser Web. Pendant le VISUALISER SELECTION, 4D permet de naviguer de façon transparente entre la page de sélection et l'affichage séparé de chaque enregistrement. 4D peut également utiliser MODIFIER SELECTION pour gérer la saisie des données et le verrouillage des enregistrements, par l'intermédiaire des formulaires Web.

(10) Pendant qu'il navigue parmi les enregistrements de la sélection, l'utilisateur clique sur un bouton dans la zone de pied de page du formulaire. La méthode objet du bouton appelle la commande ENVOYER FICHIER HTML.

(11) Cette étape est semblable aux étapes 3 et 7. Si la page HTML contient des liens, il est possible de naviguer parmi plusieurs pages. Lorsque finalement la commande ENVOYER FICHIER HTML("") est appelée, le mode HTML est terminé.

(12) L'exécution de la méthode objet du bouton sur lequel l'utilisateur a cliqué et l'afichage de la sélection liée à AJOUTER ENREGISTREMENT reprennent. Notez que les étapes (10) et (11) peuvent être répétées plusieurs fois pendant la navigation dans la sélection.

(13) Enfin, l'affichage de la sélection est terminée est le process de connexion Web est exécuté.

Et ainsi de suite...

La navigation Web libre (lorsque par exemple vous cliquez sur les boutons Suivant ou Précédent) est possible au sein de tout appel à ENVOYER FICHIER HTML (les zones vertes dans le schéma ci-dessus). En revanche, toute page HTML basée sur 4D (saisie des données, affichage de la sélection..., y compris les boîtes de dialogue standard telles que celles qui sont affichées par les commandes CONFIRMER ou Demander) échappe aux contrôles de navigation du browser. A l'issue de la navigation, 4D va si nécesssaire synchroniser les sessions Web avec celles de la base en revenant à la page Web dont le numéro de sous-contexte correspond à celui de la commande en cours d'exécution dans le process de connexion Web.

Process de connexion Web et session Web

De son point de vue, l'utilisateur pilote la session Web à travers ses actions dans le browser Web.
Du point de vue de la programmation, c'est le process de connexion Web qui pilote la session Web, et non l'inverse. Le browser Web affiche les pages envoyées par le process de connexion Web qui :

• soit exécute du code 4D,

• soit attend du browser le retour de la page Web courante.

Du point de vue du mode Structure, le process de connexion Web doit être considéré comme un process 4D dont le domaine d'exécution est 4e Dimension ou 4D Server, mais dont l'interface utilisateur est située sur le browser Web distant.

En conséquence, tenez toujours compte de la dualité du process de connexion Web lorsque vous développez des applications pour le Web, en mode contextuel. Par exemple :

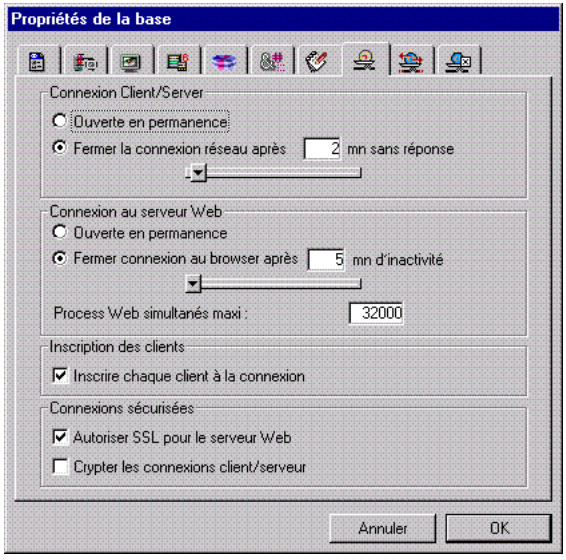
• Lors de la saisie de tout type de données, la barre de menus principale est celle du browser, pas celle de 4D. Dans un formulaire, ne comptez pas sur la barre de menus de 4D ; elle apparaît sur la machine du serveur Web, mais pas sur la machine du browser Web.

• Lorsque vous créez des formulaires à destination du browser Web, rappelez-vous que les fonctionnalités du formulaire 4D sont restreintes à celles du HTML (avec cependant quelques possibilités supplémentaires liées à 4D). Il n'est donc pas possible d'employer toutes les fonctionnalités des formulaires de 4D (par exemple, tous les types d'objets ou tous les événements formulaires). Pour plus d'informations sur ce point, référez-vous à la section Services Web, Encapsulation HTML et Javascript.

• En ce qui concerne la communication interprocess, la commande APPELER PROCESS n'a pas d'effet lorsqu'elle est appliquée à un process de connexion Web, car le formulaire actif est affiché sur le browser Web. En revanche, un process de connexion Web peut exécuter un APPELER PROCESS à destination d'un autre process 4D. De plus, la communication interprocess peut s'établir indifféremment dans les deux sens, par l'intermédiaire des commandes LIRE VARIABLE PROCESS et ECRIRE VARIABLE PROCESS, qui ne nécessitent pas la présence d'une interface utilisateur pour le process.

Temporisation de connexion Web (*Timeout*)

Comme décrit précédemment, un process de connexion Web soit exécute du code 4D, soit attend le retour de la page Web en cours d'affichage sur le browser. Dans ce second cas, le process de connexion Web accordera au browser un délai de réponse supérieur ou égal à la valeur de l'option **Fermer connexion au browser après** xx **mn d'inactivité**, que vous fixez soit dans la fenêtre des Propriétés de la base (cf. ci-dessous), soit à l'aide de la commande FIXER TEMPORISATION WEB.



La portée de ce paramétrage est la session de la base. Tous les process de connexion Web sont assujettis à cette valeur ; ils sont immédiatement affectés lorsque le paramétrage est modifié. La valeur par défaut est 5 minutes.

Note : La commande FIXER TEMPORISATION WEB vous permet de définir une valeur de *timeout* par process Web.

Vous pouvez augmenter ou réduire cette valeur à votre convenance. Par exemple, vous pouvez augmenter la temporisation si votre application permet aux utilisateurs Web de naviguer vers d'autres sites Web via des liens HTML dans les pages de votre base. En augmentant la temporisation, vous permettez aux utilisateurs de naviguer plus longtemps parmi d'autres sites Web sans fermer les connexions à votre base.

ATTENTION : Il n'est pas possible de stopper par programmation le process de connexion Web. Si vous spécifiez une temporisation longue, le process attendra ce délai même si l'utilisateur Web s'est déconnecté de la base depuis un certain temps. Si vous cochez l'option **Ouverte en permanence**, les process de connexion Web ne s'arrêteront que lorsque vous quitterez la base.
A noter toutefois qu'un process de connexion Web est automatiquement tué dès que le serveur Web passe en mode sans contexte.

Astuce : A la différence du process de serveur Web, les process de connexion Web peuvent être tués à l'aide de la commande **Tuer** (disponible dans l'Explorateur d'exécution de 4e Dimension lorsque la page Process est affichée).

Référence

FIXER PARAMETRE BASE, Méthode base Sur authentification Web, Méthode base Sur connexion Web, Services Web, Mode sans contexte, Services Web, Sécurité des connexions.

Méthode base Sur connexion Web

Serveur Web

version 6.5 (Modifiée)

La Méthode base Sur connexion Web est appelée par 4e Dimension et 4D Server chaque fois qu'un navigateur Web se connecte à la base en mode contextuel, ou qu'une requête nécessite la création d'un contexte (voir ci-dessous le paragraphe "Appels de la Méthode base Sur connexion Web").
Bien entendu, le navigateur doit auparavant avoir été "accepté" par la Méthode base Sur authentification Web (si elle existe), et la base doit être publiée en tant que serveur Web.

La Méthode base Sur connexion Web reçoit six paramètres de type Texte, passés par 4D ($1, $2, $3, $4, $5 et $6). Voici leur description :

Paramètres	Type	Description
$1	Texte	URL
$2	Texte	En-tête + corps HTTP (dans la limite des 32 ko)
$3	Texte	Adresse IP du navigateur
$4	Texte	Adresse IP appelée du serveur
$5	Texte	Nom d'utilisateur
$6	Texte	Mot de passe

Vous devez déclarer ces six paramètres de la manière suivante :

```
` Méthode base Sur connexion Web

C_TEXTE($1;$2;$3;$4;$5;$6)

` Code pour la méthode
```

• Données supplémentaires de l'URL

Le premier paramètre ($1) est l'URL saisi par l'utilisateur dans la zone 'Adresse' de son navigateur Web, moins l'adresse hôte.

Prenons l'exemple d'une connexion Intranet. Supposons que l'adresse IP de votre machine serveur Web 4D est 123.4.567.89. Le tableau suivant liste les valeurs de $1 selon l'URL saisi dans le navigateur Web :

URL saisi dans la zone Adresse du navigateur	Valeur du paramètre $1
123.4.567.89	/
http://123.4.567.89	/
123.4.567.89/Clients	/Clients
http://123.4.567.89/Clients	/Clients
http://123.4.567.89/Clients/Ajouter	/Clients/Ajouter
123.4.567.89/Faire_ceci/Si_OK/Faire_cela	/Faire_ceci/Si_OK/Faire_cela

Notez que vous êtes libre d'utiliser ce paramètre à votre convenance. 4D ignore simplement les valeurs passées au-delà de la partie hôte de l'URL.

Par exemple, vous pouvez établir une convention dans laquelle la valeur "/Clients/Ajouter" signifie "aller directement à l'ajout d'un nouvel enregistrement dans la table [Clients]." En fournissant aux utilisateurs Web de votre base une liste des valeurs possibles et/ou des marqueurs par défaut, vous pouvez proposer des raccourcis vers différentes parties de votre application. Ainsi, les utilisateurs Web peuvent accéder rapidement aux ressources de votre site Web sans devoir naviguer dans toute la base à chaque connexion.

Cas particulier : En mode sans contexte, le paramètre $1 ne débute PAS par le caractère "/" dans le cas où l'URL demandé ne correspond ni à une page existante ni à un URL spécial de 4D (tel que "/4DCGI"). Par exemple, si l'URL est "http://123.4.567.89/Clients/Ajouter" et qu'il n'existe pas de page "Ajouter" dans le dossier "Clients", la Méthode base Sur authentification Web puis la Méthode base Sur connexion Web sont appelées avec la valeur "Clients/Ajouter" dans $1.

ATTENTION : Pour empêcher un utilisateur d'accéder directement à une base à l'aide d'un marqueur créé pendant une session précédente, 4D intercepte tout URL correspondant à un des URLs standard de 4D.

• En-tête et corps de la requête HTTP

Le deuxième paramètre ($2) est l'en-tête suivi du corps de la requête HTTP envoyée par le browser Web. Notez que ces informations sont passées telles quelles à votre Méthode base Sur connexion Web. Le contenu varie en fonction du type de browser Web qui tente de se connecter.

Avec Netscape 3.0 sous Windows NT, vous recevrez un en-tête semblable à celui-ci :

```
GET HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/3.01 (WinNT; I)
Host: 192.9.200.11
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
```

Avec Microsoft Internet Explorer sous Windows NT, vous recevrez un en-tête semblable à celui-ci :

```
GET / HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Accept-Language: en
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0d; Windows NT)
Connection: Keep-Alive
If-Modified-Since: Sunday, 10-Dec-96 01:51:37 GMT
```

Si votre application exploite ces informations, il est de votre ressort d'analyser l'en-tête et le corps.

• Adresse IP du navigateur

Le troisième paramètre ($3) reçoit l'adresse IP de la machine du navigateur. Cette information peut vous permettre, en particulier, de distinguer les connexions Intranet des connexions Internet.

• Adresse IP demandée du serveur

Le quatrième paramètre ($4) reçoit l'adresse IP demandée du serveur Web 4D. En effet, à compter de la version 6.5 du programme, 4D autorise le multi-homing, permettant d'exploiter des machines disposant de plusieurs adresses IP. Pour plus d'informations sur ce point, reportez-vous à la section Services Web, Paramétrages du serveur Web.

• Nom d'utilisateur et Mot de passe

Les paramètres $5 et $6 reçoivent le nom d'utilisateur et le mot de passe saisis par l'utilisateur dans la boîte de dialogue standard d'identification affichée par le browser, le cas échéant.
Cette boîte de dialogue apparaît pour chaque connexion dès que l'option Utiliser les mots de passe est cochée, dans les Propriétés de la base (cf. section Services Web, Sécurité des connexions).

Note : Si le nom d'utilisateur envoyé par le browser existe dans 4D, pour des raisons de confidentialité le paramètre $6 n'est alors pas rempli (il reçoit une chaîne vide).

Appels de la Méthode base Sur connexion Web

La Méthode base Sur connexion Web est automatiquement appelée lorsqu'un browser Web est autorisé à se connecter à la base. Mais de plus, elle joue également le rôle de point d'entrée dans 4D depuis le mode "sans contexte" du serveur Web.
En effet, le passage du mode sans contexte au mode contextuel s'effectue par l'intermédiaire de la Méthode base Sur connexion Web (pour plus d'informations sur ce point, reportez-vous à la section Services Web, Mode sans contexte).

Attention : En mode sans contexte, l'appel d'une commande 4D affichant un élément d'interface (ALERTE, DIALOGUE...) entraîne l'arrêt du traitement.

La Méthode base Sur connexion Web est donc appelée dans différents cas :

• au moment de la connexion d'un navigateur au serveur Web 4D démarrant en mode contextuel (mode standard). La méthode base est appelée avec l'URL /<action>...

• au moment du passage du mode sans contexte au mode contextuel. La méthode base est appelée avec l'URL /4DMETHOD/NomMéthode.

• lorsque 4D est appelé par une page semi-dynamique, par l'intermédiaire de l'URL /4DCGI/<action>. La méthode base est appelée avec l'URL (cf. section Services Web, Mode sans contexte).

• lorsqu'une page Web appelée en mode sans contexte avec un URL du type <chemin>/<fichier> n'est pas trouvée. La méthode base est appelée avec l'URL (*).

• lorsqu'une page Web est appelée en mode sans contexte avec un URL du type <chemin>/ et qu'aucune page d'accueil par défaut n'est définie. La méthode base est appelée avec l'URL (*).

(*) Dans ces cas particuliers, l'URL reçu dans $1 ne débute pas par le caractère "/".

Pour connaître l'origine de l'appel de la Méthode base Sur connexion Web et adapter le traitement à effectuer, vous devez utiliser la fonction Contexte Web, qui retourne Vrai si elle est appelée depuis le mode contextuel, et Faux sinon.

Par conséquent, lorsque votre serveur Web manipule le mode contextuel et le mode sans contexte, nous vous conseillons de structurer la Méthode base Sur connexion Web de la manière suivante :

```
   `Méthode base Sur connexion Web
C_TEXTE($1;$2;$3;$4;$5;$6)
Si (Contexte Web)  `Si l'on est en mode contextuel
   AvecContexte ($1;$2;$3;$4;$5;$6)
      `La méthode AvecContexte contient tout ce qu'il y avait dans la
      `méthode base Sur connexion Web en 4D 6.0.x
Sinon
   SansContexte ($1;$2;$3;$4;$5;$6)
      `La méthode SansContexte effectue les traitements des requêtes
      `non contextuelles (généralement courts)
Fin de si
```

Exemple : implémentation des pages Home locales des clients

Dans l'exemple suivant, le paramètre $1, envoyé à la méthode base Sur connexion Web, est utilisé pour implémenter les pages Home des clients dans une entreprise.

La base contient deux tables : [Clients] et [Tables]. La Méthode base Sur ouverture ci-dessous initialise les tableaux interprocess utilisés ultérieurement par la Méthode base Sur connexion Web.

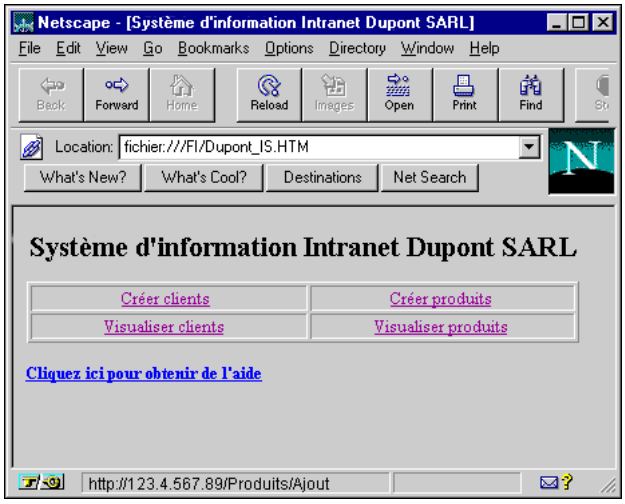
```
   ` Méthode base Sur ouverture

   ` Liste des tables
TABLEAU ALPHA(31;◊taTables;Nombre de tables)
Boucle (§vlTable;1;Taille tableau(◊taTables))
   ◊taTables{§vlTable}:=Nom de la table(§vlTable)
Fin de boucle

   ` Actions Web standard à la connexion
TABLEAU ALPHA(31;◊taActions;2)
◊taActions{1}:="Ajouter"
◊taActions{2}:="Visualiser"
```

Le principal objectif de la méthode base Sur connexion Web est de déchiffrer les données de l'URL situées après l'adresse de l'hôte et d'agir en conséquence. La méthode est la suivante :

```
   ` Méthode base Sur connexion Web

C_TEXTE($1;$2;$3;$4;$5;$6)
C_TEXTE($vtURL)

Si (Contexte Web)   ` Si nous sommes bien en mode standard
      ` Par précaution, vérifiez que $1 est égal à "/" ou "/..."
   Si ($1="/@")
         ` Copier l'URL dans une variable locale moins le premier "/"
      $vtURL:=Sous chaine($1;2)
         ` Analyser l'URL et remplir un tableau local avec les informations de l'URL
         ` Par exemple, si les données supplémentaires de l'URL sont "aaa/bbb/ccc", le
         ` tableau contiendra les trois éléments "aaa", "bbb" et "ccc", dans cet ordre
      §vlElem:=0
      TABLEAU TEXTE($atTokens;§vlElem)
      Tant que ($vtURL # "")
         §vlElem:=§vlElem+1
         INSERER LIGNES($atTokens;§vlElem)
         §vlPos:=Position("/";$vtURL)
         Si (§vlPos>0)
            $atTokens{§vlElem}:=Sous chaine($vtURL;1;§vlPos-1)
            $vtURL:=Sous chaine($vtURL;§vlPos+1)
```

```
            Sinon
               $atTokens{$vlElem}:=$vtURL
               $vtURL:=""
            Fin de si
         Fin tant que
            ` Si des données supplémentaires sont passées au-delà de la partie hôte de l'URL

         Si ($vlElem>0)
               ` Utiliser le tableau interprocess initialisé dans la méthode base Sur ouverture
               ` Vérifier si la première information est un nom de table
            $vlTableNumber:=Chercher dans tableau(◊taTables;$atTokens{1})
            Si ($vlTableNumber>0)
                  ` Si oui, obtenir un pointeur vers cette table
               $vpTable:=Table($vlTableNumber)
                  ` Définir les formulaires entrée et sortie
               FORMULAIRE ENTREE($vpTable->;"Input Web")
               FORMULAIRE SORTIE($vpTable->;"Output Web")
                  ` Utiliser le tableau interprocess initialisé dans la méthode base Sur début
                  ` Vérifier si la deuxième information est une action standard connue
               $vlAction:=Chercher dans tableau(◊taActions;$atTokens{2})
               Au cas ou
                     ` Ajouter des enregistrements
                  : ($vlAction=1)
                     Repeter
                        AJOUTER ENREGISTREMENT($vpTable->;*)
                     Jusque (OK=0)
                  : ($vlAction=2)   ` Visualiser les enregistrements
                     LECTURE SEULEMENT($vpTable->)
                     TOUT SELECTIONNER($vpTable->)
                     VISUALISER SELECTION($vpTable->;*)
                     LECTURE ECRITURE($vpTable->)
                  Sinon
                     ` Ajouter éventuellement ici des actions standard supplémentaires
               Fin de cas
            Sinon
               ` Ajouter éventuellement ici d'autres actions de table standard
            Fin de si
         Fin de si
      Fin de si
         ` Dans tous les cas, poursuivre le processus standard de connexion
      WWW CONNEXION STANDARD
   Sinon
      ... ` Placer éventuellement ici le code gérant le mode sans contexte
   Fin de si
```

A ce stade, les membres de l'entreprise peuvent se connecter à la base et saisissent un URL en fonction des conventions définies. Les utilisateurs peuvent également créer des marqueurs s'ils ne veulent pas saisir l'URL à chaque connexion. En fait, la meilleure solution consiste à fournir à chaque membre de l'entreprise une page HTML qu'il peut utiliser en local pour accéder à la base. Voici la page HTML :



Autrement dit, la page HTML Dupont_IS.HTM est la page Home du client local pour le système d'informations de l'entreprise basé sur 4D. Si l'utilisateur clique sur le lien Créer produits, le browser Web se connectera à l'hôte dont l'URL est http://123.4.567.89/Produits/Ajout. Si l'adresse IP de l'ordinateur où se trouve la base est 123.4.567.89, la méthode base Sur connexion Web reçoit les données d'URL supplémentaires "/Produits/Ajout" dans $1 puis lance le processus d'ajout d'enregistrements à la table [Produits].

Enfin, les utilisateurs peuvent glisser-déposer les liens de cette page vers leur bureau pour créer une icône de raccourci Internet, telle que l'icône ci-dessous. Double-cliquer sur une de ces icônes envoie directement l'utilisateur dans la zone spécifiée de votre base 4D Web.

Le code source de cette page HTML est le suivant :

```
<HTML>
<HEAD>
    <TITLE>Système d'information Intranet Dupont SARL</TITLE>
</HEAD>
<BODY>
<H1><B>Système d'information Intranet Dupont SARL</B></H1>
<P ALIGN=CENTER><TABLE BORDER=1 CELLPADDING=1 WIDTH="100%">
    <TR>
    <TD>
        <P ALIGN=CENTER><A HREF="http://123.4.567.89/Clients/Ajout">Créer clients</A>
    </TD>
    <TD>
        <P ALIGN=CENTER><A HREF="http://123.4.567.89/Produits/Ajout">Créer produits</A>
    </TD>
    </TR>
    <TR>
    <TD>
        <P ALIGN=CENTER><A HREF="http://123.4.567.89/Clients/Liste">Visualiser clients</A>
    </TD>
    <TD>
        <P ALIGN=CENTER><A HREF="http://123.4.567.89/Produits/Liste">Visualiser produits</A>
    </TD>
    </TR>
</TABLE></P>
<P><B><A HREF="Help.HTM">Cliquez ici pour obtenir de l'aide</A></B></P>
</BODY>
</HTML>
```

Référence

Méthode base Sur authentification Web, Présentation des méthodes base, Services Web, Mode sans contexte, Services Web, URLs et actions de formulaires.

Vous pouvez personnaliser le déroulement des sessions Web, en fonction de vos besoins. Les paramétrages suivants sont disponibles :

• Définition d'une page d'accueil (Home) par défaut.
• Utilisation de Javascript pour les contrôles de saisie.
• Mode d'insertion des variables 4D dans les pages statiques
• Modification du mode de référencement des contextes.
• Paramétrage de la conversion des caractères HTML et des filtres Web.
• Utilisation d'un cache pour les pages statiques.
• Désignation de l'adresse IP sur laquelle le serveur reçoit les requêtes HTTP.
• Définition du nombre maximum de process Web simultanés.

Page d'accueil par défaut

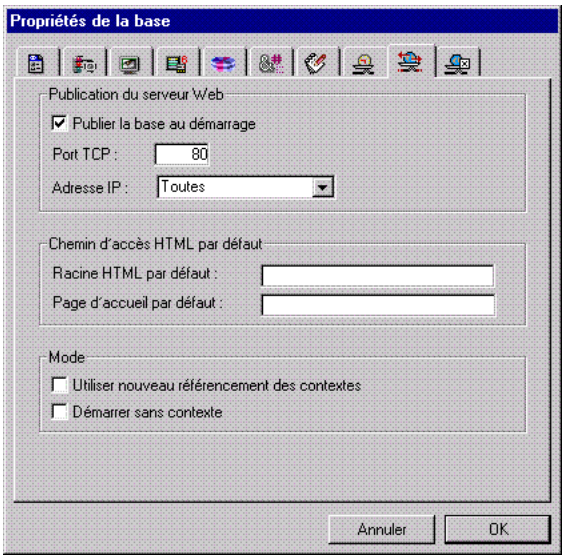
Vous pouvez définir une page d'accueil (page "Home") par défaut pour tous les browsers se connectant à la base. Cette page peut être statique ou "semi-dynamique".

Si vous spécifiez une page d'accueil par défaut, cette page est envoyée à chaque browser se connectant à la base, quel que soit le mode (avec ou sans contexte) défini pour le démarrage des sessions Web. Par conséquent, à la différence des versions de 4D antérieures à la 6.5, en mode contextuel la barre de menus courante n'est pas envoyée au browser.

Par défaut, aucune page d'accueil n'est définie. Si vous ne spécifiez pas de page d'accueil personnalisée, le comportement du serveur Web diffère suivant le mode de démarrage :
• Si le serveur Web démarre en mode contextuel (mode standard), la barre de menus courante — par défaut, la barre de menus n° 1 — est envoyée en tant que page d'accueil.
• Si le serveur Web démarre en mode sans contexte, seule la Méthode base Sur connexion Web est appelée. Il vous revient alors de traiter la requête par programmation.

Pour définir une page d'accueil par défaut :

1. Cliquez sur l'onglet "Serveur Web I" dans la boîte de dialogue des Propriétés de la base. La page suivante apparaît :



2. Dans la zone "Page d'accueil par défaut", saisissez le chemin d'accès relatif du dossier que vous souhaitez définir.
Le chemin d'accès saisi dans cette boîte de dialogue est relatif : il est établi à partir du dossier contenant la structure de la base. Afin d'assurer la compatibilité multi-plate-forme de vos bases, le serveur Web 4D utilise, pour décrire les chemins d'accès, des conventions d'écriture particulières. Les règles de syntaxe sont les suivantes ;
• les dossiers sont séparés par le caractère /
• le chemin ne doit pas se terminer par /
• pour "remonter" d'un niveau dans la hiérarchie des dossiers, saisissez ".." (point point) devant le nom du dossier
• le chemin ne doit pas commencer par /.
Par exemple, si vous souhaitez que la page d'accueil par défaut soit la page "MyHome.htm", placée dans le dossier "Web", saisissez Web/MyHome.htm

3. Validez la boîte de dialogue.

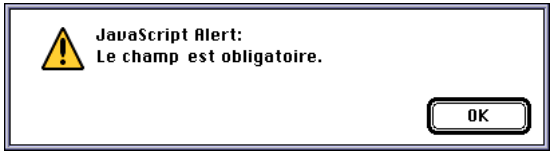
Note : Vous pouvez également définir une page d'accueil par défaut pour chaque process Web à l'aide de la routine FIXER PAGE ACCUEIL.

Utiliser Javascript pour les contrôles de saisie

Une partie des contrôles de saisie peut être effectuée sur les browsers via des scripts Java automatiques.

Sur le browser, les contrôles de saisie et les types de données (champs ou variables) auxquels ils peuvent s'appliquer sont les suivants :
• valeur minimale (numériques) ;
• valeur maximale (numériques) ;
• valeur obligatoire (numériques et alphas).

Les scripts Java générés, de petite taille, ont pour but d'afficher les boîtes de dialogue d'alerte adéquates sans réellement empêcher la validation (celle-ci reste du ressort de 4D). En effet, si une zone de saisie contient une valeur incorrecte, un message d'alerte est affiché sur le browser lorsque l'utilisateur clique sur un bouton (de validation, d'annulation, etc.) :



Une fois la boîte de dialogue d'alerte validée, si l'utilisateur clique une seconde fois sur le bouton, l'action de celui-ci sera prise en compte.
Le contrôle complet de la saisie est effectué sur le serveur Web, en mode Utilisation ou en Menus créés.

Pour activer les contrôles de saisie Javascript :

1. Dans la boîte de dialogue des Propriétés de la base, cliquez sur l'onglet "Serveur Web II".

2. Cochez l'option "Utiliser Javascript pour les contrôles de saisie".
Par défaut, cette option n'est pas cochée.

3. Cliquez sur le bouton **OK**.

Mode d'insertion des variables 4D dans les pages statiques

L'option **Utiliser les commentaires 4DVAR au lieu des crochets**, située dans la page "Serveur Web II" des Propriétés de la base, vous permet de définir la notation à utiliser pour l'insertion de variables 4D dans les pages statiques.
• Lorsque l'option est cochée, la syntaxe à employer est la notation HTML standard : <!--4DVAR MAVAR--> (l'espace entre 4DVAR et le nom de la variable est impératif).
• Lorsque l'option n'est cochée (valeur par défaut), la syntaxe à employer est la notation avec les crochets ([MAVAR]) — qui est une solution propriétaire.

Mode de référencement des contextes

Par défaut, à chaque action de l'utilisateur, le serveur Web 4D envoie au browser le numéro du contexte courant. C'est-à-dire que, si une page contient deux paragraphes et une image, 4D enverra trois fois le numéro de contexte.
Il est possible de modifier ce fonctionnement, ce qui permet d'accélérer l'envoi des pages : avec le nouveau mode de référencement des contextes, le numéro de contexte est placé dans l'URL de base du document. Ainsi, il n'est plus répété pour chaque objet.

Pour exploiter le nouveau mode de référencement des contextes :

1. Dans la page "Serveur Web I" des Propriétés de la base, cochez l'option **Utiliser nouveau référencement des contextes**.
Par défaut, cette case n'est pas cochée.

2. Cliquez sur le bouton **OK** puis redémarrez la base afin que le nouveau fonctionnement soit effectif.

Envoi direct des caractères ASCII étendus

Par défaut, le serveur Web 4D convertit les caractères ASCII étendus présents dans les pages Web (dynamiques et statiques) au normes HTML avant de les envoyer. Ils sont ensuite interprétés par les browsers.
Vous pouvez paramétrer le serveur Web de manière à ce que les caractères ASCII étendus soient envoyés "tel quels", sans conversion en entités HTML. Cette option permet un gain de vitesse important sur des systèmes étrangers (principalement japonais).

Pour que le serveur Web 4D envoie directement les caractères ASCII étendus :

1. Dans la boîte de dialogue des Propriétés de la base, cliquez sur l'onglet "Serveur Web II".

2. Cochez l'option **Envoyer directement les caractères étendus**.

3. Validez la boîte de dialogue.

Edition des tables de conversion des caractères depuis 4D

Vous pouvez modifier directement depuis 4D les tables de conversion des caractères ASCII (filtres Web) en entrée et en sortie. Auparavant, cette fonction était disponible par l'intermédiaire de l'utilitaire Customizer Plus.
La modification des tables de conversion n'est possible que lorsque le jeu de caractères "x-user-defined" est sélectionné dans la boîte de dialogue des Propriétés de la base.

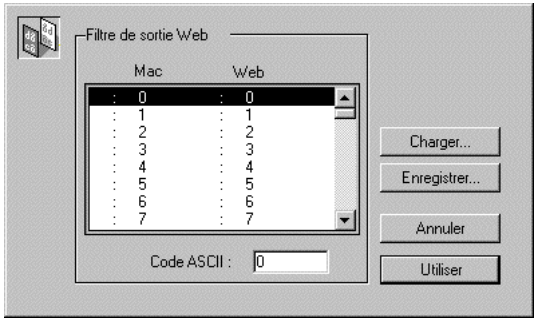
Pour modifier les tables de conversion des caractères ASCII :

1. Dans la boîte de dialogue des Propriétés de la base, cliquez sur l'onglet "Serveur Web II".

2. Dans la liste déroulante "Jeu de caractères", choisissez l'option **x-user-defined**.

Les boutons **Editer filtre de saisie** et **Editer filtre de sortie** sont alors actifs.

3. Cliquez sur le bouton correspondant au filtre que vous souhaitez modifier.
Le filtre de saisie interprète les caractères envoyés par le browser au serveur Web 4D. Le filtre de sortie interprète les caractères envoyés par le serveur Web 4D.
La boîte de dialogue d'édition des filtres Web apparaît :



Note : Cette boîte de dialogue est semblable à la boîte d'édition des filtres d'import/export de 4D. Toutefois, les deux fonctions sont indépendantes : les filtres Web ne sont PAS liés aux filtres d'import/export.

4. Dans la zone de défilement, recherchez et cliquez sur le caractère Mac que vous souhaitez filtrer.
OU
Chargez un filtre Web déjà créé et sauvegardé en cliquant sur le bouton **Charger** (allez ensuite à l'étape n° 8).

5. Dans la zone de saisie "Code ASCII", saisissez le nouveau code ASCII du caractère.

6. Répétez l'opération pour tous les caractères à filtrer.

7. Si vous le souhaitez, cliquez sur le bouton **Enregistrer** pour sauvegarder le filtre.

8. Cliquez sur le bouton **Utiliser**.

9. Cliquez sur le bouton **OK** de la boîte de dialogue des Propriétés de la base.
Le filtre de saisie et/ou de sortie Web est alors actif.

Cache pour les pages statiques

Le serveur Web 4D dispose d'un cache permettant de charger en mémoire les pages statiques, les images GIF, les images JPEG (<100 ko) et les feuilles de styles (fichiers .css), au fur et à mesure qu'elles sont demandées.
L'utilisation d'un cache permet d'augmenter de manière significative les performances du serveur Web, en ce qui concerne l'envoi de page statiques.

Le cache est commun à tous les process Web. Vous pouvez fixer sa taille dans les Propriétés de la base. Par défaut, le cache des pages statiques n'est pas activé (sa taille vaut 0).

Pour activer le cache des pages statiques :

1. Dans la page “Serveur Web II” des Propriétés de la base, fixez une valeur (exprimée en Ko) pour le cache des pages statiques.

La valeur à fixer dépend du nombre et de la taille des pages statiques de votre site Web, ainsi que des ressources dont dispose la machine hôte.

Note : Au cours de l’utilisation de votre base Web, vous pourrez contrôler les performances du cache à l’aide de la routine STATISTIQUES DU CACHE WEB. Si par exemple vous constatez que le taux d’utilisation du cache est proche de 100%, vous pouvez envisager d’augmenter la taille qui lui est allouée.
Les URL particuliers /4DSTATS et /4DHTMLSTATS vous permettent également d’obtenir des informations sur l’état du cache. Reportez-vous à la section Services Web, Informations sur le site Web.

2. Cliquez sur **OK**.
Une fois le cache activé, lorsqu’une page est demandée par un browser, le serveur Web 4D la cherche d’abord dans le cache. Si elle s’y trouve, elle est immédiatement envoyée, sinon le programme charge la page depuis le disque et la place dans le cache.
Lorsque le cache est plein et que de la place supplémentaire est requise, 4D “décharge” les pages les moins utilisées, par ordre d’ancienneté.

Vider le cache

Vous pouvez à tout moment vider le cache des pages et des images qu’il contient (par exemple si vous avez effectué des modifications sur une page statique et souhaitez qu’elle soit rechargée dans le cache).
Pour cela, il vous suffit de cliquer sur le bouton **Vider cache** dans la page “Serveur Web II” des Propriétés de la base. Le cache est alors immédiatement vidé.

Désigner l’adresse IP pour les requêtes HTTP

Dans la page “Serveur Web I” des Propriétés de la base, il est possible de définir l’adresse IP sur laquelle le serveur Web doit recevoir les requêtes HTTP.
Par défaut, le serveur répond sur toutes les adresses IP (option **Toutes**).

Le pop up menu “Adresse IP” liste automatiquement toutes les adresses IP présentes sur la machine. Lorsque vous sélectionnez une adresse particulière, le serveur ne répond qu’aux requêtes dirigées sur cette adresse.

Cette fonctionnalité est destinée aux serveurs Web 4D hébergés sur des machines ayant plusieurs adresses TCP/IP. C'est, par exemple, fréquemment le cas chez les fournisseurs d'hébergement Internet.

Note : La mise en place de cette fonctionnalité nécessite une configuration adéquate de la machine accueillant les différents serveurs Web. Pour plus d'informations sur ce point, reportez-vous à la section Gestion d'adresses IP multiples (Web) dans le manuel *Composants réseau pour 4D Server*.

Définir le nombre maximum de process Web

La page "Connexions" des Propriétés de la base vous permet de définir la limite strictement supérieure du nombre de Process Web simultanés maxi. Ce paramètre permet de prévenir la saturation du serveur Web 4D pouvant se produire lors d'un envoi massif de requêtes ou d'une demande excessive de création de contextes.

Ce paramètre définit le nombre maximum de process Web de tout type : contextuels, non contextuels ou appartenant à la "réserve" de process. Par défaut, ce nombre est de 32 000 (autrement dit, jusqu'à 31 999 process Web peuvent être créés simultanément). Vous pouvez passer toute valeur incluse entre 10 et 32 000.

Lorsque ce nombre maximum (moins un) de process Web concurrents est atteint, 4D ne crée plus de nouveau process et retourne le message "Serveur non disponible" (statut HTTP 503 - Service Unavailable) à toute nouvelle requête.

Note : Le nombre maximum de process Web peut également être défini à l'aide de la commande FIXER PARAMETRE BASE.

Comment déterminer la valeur à passer ?

En théorie, le nombre maximum de process Web est le résultat de la division Mémoire disponible / Taille de la pile d'un process Web.
Une autre solution consiste à visualiser les informations sur les process Web affichées dans l'Exporateur d'exécution : le nombre courant de process Web et le nombre maximum atteint depuis le démarrage du serveur Web sont indiqués.

A propos de la réserve de process Web

La "réserve" de process Web permet d'augmenter la réactivité du serveur Web en mode sans contexte. Cette réserve est dimensionnée par un minimum (0 par défaut) et un maximum (10 par défaut) de process à recycler. Ces valeurs peuvent être modifiées à l'aide de la commande FIXER PARAMETRE BASE (sélecteurs 6 et 7). Lors du changement du nombre maximum de process Web, si celui-ci est inférieur à la limite supérieure de la "réserve", cette limite est alors abaissée au nombre maximum de process Web.

Référence

FIXER PAGE ACCUEIL, FIXER PARAMETRE BASE, Services Web, Mode sans contexte, Services Web, Process de connexion Web, Services Web, Sécurité des connexions, Services Web, Support du HTML.

Le serveur Web de 4D (depuis la version 6.5) peut être utilisé en **mode sans contexte**. 4D devient dans ce mode un serveur HTTP “classique”.

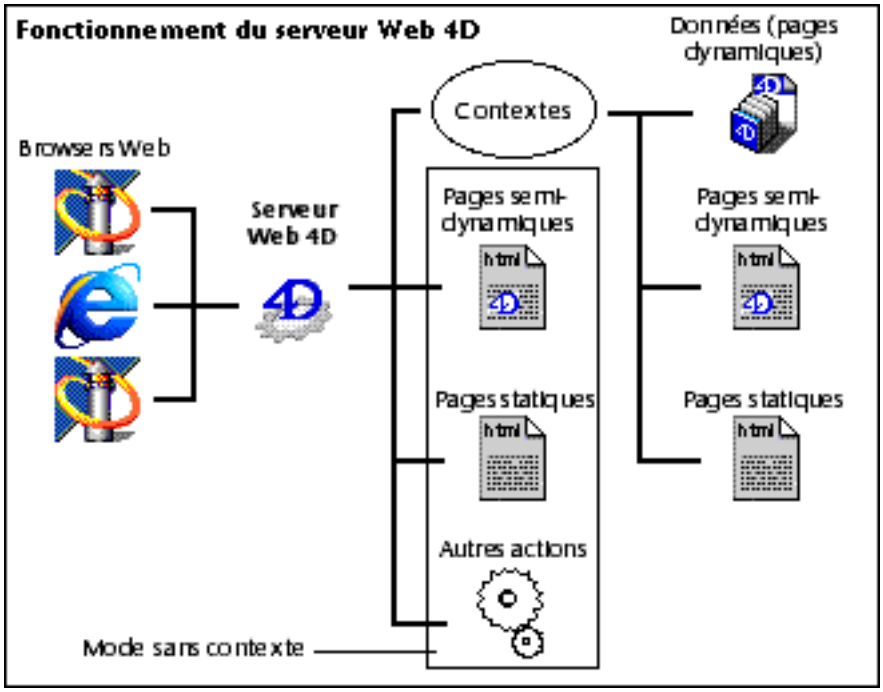
Mode contextuel et mode sans contexte

Par défaut, les fonctionnalités Web de 4D utilisent la notion de **contextes** : un browser Web se connectant à la base crée un contexte, dans lequel il dispose de sa propre sélection courante, de ses variables, etc. En quelque sorte, chaque browser est considéré comme un 4D Client.
Ce système permet au serveur Web 4D de contrôler parfaitement les actions des browsers et de garantir l’intégrité des données. Il s’avère toutefois peu adapté lorsque 4D est utilisé comme serveur HTTP “classique”, c’est-à-dire lorsque 4D envoie des pages HTML statiques. Dans ce cas, le contexte généré par le browser reste actif jusqu’à ce que la période d’inactivité spécifiée (*timeout*) soit atteinte. Si par exemple le browser a quitté le site entre-temps, le contexte est alors “gaspillé”. De plus, certaines fonctions standard des browsers (**Recharger**, **Page précédente**, etc.) ne sont pas utilisables (cf. section Services Web, Process de connexion Web).

Pour vous permettre d’utiliser le serveur Web 4D en tant que serveur HTTP classique lorsqu’il envoie des pages statiques, 4D vous propose le ***mode sans contexte***. Dans ce mode, le serveur Web 4D devient alors un serveur HTTP parfaitement standard : les pages Web statiques sont envoyées sans qu’il soit nécessaire de maintenir de contexte.
Ces pages statiques peuvent alors bénéficier d’un cache réglable (cf. section Services Web, Paramétrages du serveur Web). Dans ce cas également, vous pouvez obtenir des statistiques sur la consultation de ces pages grâce à la routine STATISTIQUES DU CACHE WEB.

Bien entendu, les contextes sont toujours utilisés pour contrôler la navigation des browsers parmi les enregistrements. En fait, vous pouvez utiliser le serveur Web 4D dans le mode que vous souhaitez : mode contextuel, mode sans contexte, ou passer à la volée d’un mode à l’autre en fonction de vos besoins.

Le principe de fonctionnement du serveur Web 4D est résumé dans le schéma suivant :



Passer d'un mode à l'autre

Par défaut, le serveur Web de 4D fonctionne de la même manière que dans les versions 6.0.x, en ce qui concerne la gestion des contextes : un contexte est généré à chaque connexion Web. Il vous appartient d'indiquer explicitement à 4D que vous souhaitez passer en mode sans contexte.

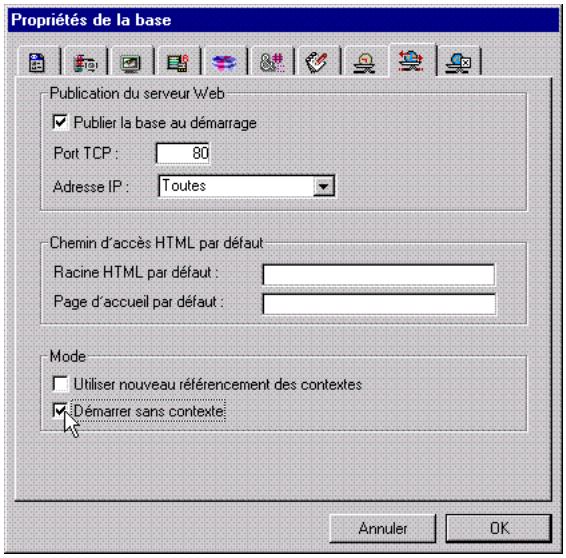
Définir le mode sans contexte au démarrage

Vous pouvez démarrer le serveur Web directement en mode sans contexte. Cela signifie que, lorsqu'un utilisateur se connecte à la base, aucun contexte n'est généré.

Pour définir le mode sans contexte au démarrage :

1. Dans la boîte de dialogue des Propriétés de la base, cliquez sur l'onglet "Serveur Web II".

2. Cochez l'option **Démarrer sans contexte** :



Par défaut, l'option n'est pas cochée : le serveur Web démarre en mode contextuel.
3. Validez la boîte de dialogue.
Le serveur Web de 4D envoie alors la page d'accueil en mode sans contexte.
• Si vous avez défini une page d'accueil personnalisée dans la zone "Page d'accueil par défaut" des Propriétés de la base (cf. section Services Web, Paramétrages du serveur Web), la page est envoyée.

Note : Si l'URL demandé se termine par /, le nom de la page d'accueil lui est ajouté et 4D tente de l'envoyer. Par exemple, si l'URL demandé est /Dossier/ et que la page d'accueil par défaut est "MaPage.HTM", 4D cherchera /Dossier/MaPage.HTM. S'il ne la trouve pas, la Méthode base Sur connexion Web est appelée.

• Si vous n'avez pas défini de page d'accueil par défaut, ou si la page d'accueil définie n'existe pas, la Méthode base Sur connexion Web est appelée (en mode sans contexte). Vous pouvez alors traiter la connexion de manière personnalisée. Par exemple, vous pouvez utiliser la commande FIXER PAGE ACCUEIL pour choisir une page d'accueil en fonction de l'utilisateur connecté.

Passer du mode contextuel au mode sans contexte

Au cours de l'exploitation de la base, vous pouvez à tout moment passer du mode contextuel au mode sans contexte, et inversement.

Pour que le serveur Web 4D passe en mode sans contexte, vous disposez de plusieurs solutions :

• Exécuter la commande ENVOYER BLOB HTML ou ENVOYER TEXTE HTML en passant Vrai dans le paramètre optionnel (sansContexte). Par exemple : ENVOYER BLOB HTML (MonBlob;".HTM";Vrai)

4D envoie alors les données en mode sans contexte. Le process gérant le contexte est immédiatement tué.

Note : Ne pas passer le dernier paramètre équivaut à le mettre à Faux. Si le mode sans contexte est déjà actif, le paramètre est ignoré. Vous pouvez connaître le mode courant à l'aide la fonction Contexte Web.

• Exécuter la commande ENVOYER REDIRECTION HTTP.
Cette commande permet de rediriger la requête vers l'URL passé en paramètre. Appelée depuis le mode contextuel, elle provoque la fermeture du process Web.

• Appeler un URL débutant par /4DACTION (cf. paragraphe "Utiliser le mode sans contexte" ci-dessous).
Le mode sans contexte reste utilisé tant qu'aucun URL activant le mode contextuel n'est appelé.

Passer du mode sans contexte au mode contextuel

Schématiquement, le mode contextuel "classique" de 4D est utilisé dès que la base de données est sollicitée, c'est-à-dire dès que le serveur Web doit générer des pages dynamiques.
L'appel à la base de données 4D depuis le mode sans contexte s'effectue par l'intermédiaire de l'URL particulier : /4DMETHOD/MaMéthode. Il vous suffit de placer cet URL dans une page statique. Pour plus d'informations sur cet URL, reportez-vous aux sections Encapsulation HTML et Javascript et Services Web, URLs et actions de formulaires.

Lorsque cet URL est activé, le serveur Web 4D crée un nouveau contexte et effectue les opérations suivantes :
• la Méthode base Sur authentification Web est exécutée (si elle existe),
• la Méthode base Sur connexion Web est exécutée (si elle existe) — dans ce cas particulier, $1 est égal à /4DMETHOD/MaMéthode au lieu de / (barre oblique),
• enfin, la méthode demandée est exécutée dans le contexte nouvellement créé.
Le fonctionnement du serveur Web est alors standard. Le mode contextuel reste utilisé tant qu'aucune commande activant le mode sans contexte n'est appelée.

Utiliser le mode sans contexte

En mode sans contexte, 4D ne maintient par définition aucune variable, sélection, etc., liée à la navigation du browser. Celle-ci s'effectue de manière entièrement libre, sans que 4D soit informé.

Toutefois, le serveur Web 4D propose différents outils permettant de construire, gérer et envoyer des pages Web ou des données HTML dont le contenu est en totalité ou en partie issu d'un traitement effectué par 4D. Vous pouvez envoyer en mode sans contexte des **pages semi-dynamiques**. Les pages semi-dynamiques ont pour but de vous permettre de tirer parti de 4D au sein de vos pages Web tout en conservant les avantages liés au mode sans contexte. Les pages semi-dynamiques sont utilisables en mode contextuel et en mode sans contexte. Toutefois, elles sont principalement destinées au mode sans contexte.

En mode sans contexte, vous pouvez également lire et écrire les champs des en-têtes HTTP des requêtes et de leurs reponses, ce qui vous permet de gérer les "cookies".

Note : En revanche, certaines fonctions du serveur Web 4D ne sont pas disponibles en mode sans contexte. Dans cette documentation, la compatibilité des fonctions du serveur Web avec les modes contextuel et sans contexte est décrite au cours du texte.

Pour plus d'informations sur les fonctions disponibles en mode sans contexte, reportez-vous aux sections suivantes :
- Encapsulation HTML et Javascript
- Services Web, URLs et actions de formulaires
- Services Web, Balises HTML 4D

Envoyer des fichiers ".shtm" aux browsers

Lorsque le serveur Web 4D envoie une page HTML dont l'extension est ".shtm" (ou ".shtml"), il analyse les références aux variables 4D et les balises HTML telles que 4DSCRIPT et 4DVAR (cf. Services Web, Balises HTML 4D) éventuellement présentes dans la page, avant de l'envoyer au browser.
Ce fonctionnement vous permet, en mode sans contexte, d'effectuer des traitements dans 4D avant d'afficher une page HTML statique tout en assurant simplement la mise à jour de références dynamiques qu'elle contient. Veillez toutefois à effectuer les traitements les plus courts possibles, afin de ne pas bloquer le browser.
Un exemple d'utilisation de ce type de page est fourni dans la description de la routine STATISTIQUES DU CACHE WEB.

Connaître le nombre de contextes générés

En fonction des actions qu'ils effectuent, certains process Web utilisent des contextes Web, d'autres n'en utilisent pas.
Vous pouvez connaître le nombre de contextes générés, à l'aide de la commande INFORMATIONS PROCESS : pour tout process Web, cette commande indique dans le paramètre origine s'il utilise un contexte (-11, Process web avec contexte) ou non (-3, Process web sans contexte).

Référence

Contexte Web, Encapsulation HTML et Javascript, ENVOYER BLOB HTML, ENVOYER REDIRECTION HTTP, LIRE VARIABLES FORMULAIRE WEB.

Avant d'implémenter du code HTML et du JavaScript dans votre application 4D, nous vous conseillons d'examiner ce que réalise automatiquement 4e Dimension.

Barres de menus

• Chaque barre de menu est transcrite dans une page HTML. Chaque menu apparaît comme du texte seul et les commandes de menu apparaissent comme des liens vers les méthodes 4D.
• Une image associée à une barre de menus est placée au-dessous des menus sur le browser.
• Cliquer sur une commande de menu dans le browser Web déclenche, dans le process de connexion Web, l'exécution de la méthode 4D associée.

Formulaires

• 4D conserve au maximum la position initiale de chaque objet du formulaire. Notez cependant que le HTML est une application orientée traitement de texte : le positionnement relatif des objets pourra être légèrement différent sur le browser.

• Les formulaires multi-pages (y compris la page zéro et les formulaires hérités) sont gérés de manière transparente.

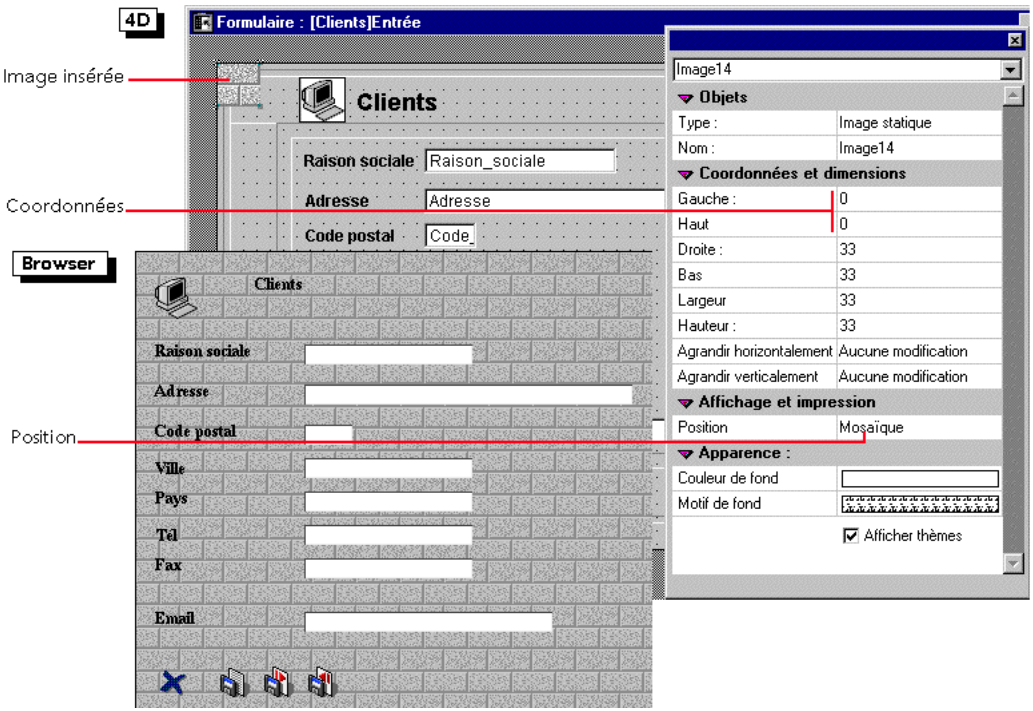
• Les actions automatiques, lorsqu'elles sont cohérentes, sont supportées de manière transparente.

• Les événements formulaires Sur chargement, Sur libération, Sur clic souris et Sur minuteur sont gérés. Les autres événements ne le sont pas.

• Les balises d'en-tête, de corps, de rupture et de pied de page sont prises en compte lors des appels à VISUALISER SELECTION et MODIFIER SELECTION. L'en-tête du formulaire s'affiche une fois au début de la page HTML, le corps est répété autant de fois que nécessaire et les variables (telles que les boutons) sont placées dans le pied de page à la fin de la page HTML, juste en-dessous des liens automatiques de navigation dans la sélection.

• Les info-bulles des boutons affichés en images dans l’éditeur de formulaires apparaissent sur le browser — si celui-ci en permet l’affichage.

• Une image répliquée (position “Mosaïque”) insérée en coordonnées (0,0,x,x) dans l’éditeur de formulaires de 4D est envoyée comme image de fond au browser :



A noter toutefois que les images sombres sont à éviter.

Note : Le serveur Web 4D utilise les CSS1 (*Cascading Style Sheet* 1) pour générer des pages HTML dont l’apparence est très proche de celle des formulaires obtenus dans 4e Dimension. Ces "feuilles de style en cascade", définies par le consortium W3C (World Wide Web Consortium), contiennent un ensemble de caractéristiques définissant l’apparence d’un document : police de caractères, taille, couleur, titres, texte courant, interlignage, etc. Depuis la version 6.5 de 4D, les documents .CSS sont envoyés avec le type MIME “text/css”. En mode contextuel et en mode sans contexte, ces documents ne sont pas analysés par 4D.
Pour des raisons de compatibilité, vous pouvez choisir, à l’aide de la commande FIXER PARAMETRE BASE (sélecteur 8, Mode conversion Web), le mode de conversion Web à utiliser pour les formulaires.

Champs

Lorsqu'un formulaire 4D est converti en page HTML, les champs sont traduits de la manière suivante :

Type champ 4D	Objet HTML	Balise HTML
Alphanumérique	Champ texte (*)	<INPUT Type="text" ...>
Texte	Champ texte (*)	<TEXTAREA ...> (**) <INPUT Type="text" ...> (***)
Réel	Champ texte (*)	<INPUT Type="text" ...>
Entier	Champ texte (*)	<INPUT Type="text" ...>
Entier long	Champ texte (*)	<INPUT Type="text" ...>
Date	Champ texte (*)	<INPUT Type="text" ...>
Heure	Champ texte (*)	<INPUT Type="text" ...>
Booléen	Bouton radio ou case à cocher (*)	<INPUT Type="radio" ...> <INPUT Type="checkbox" ...>
Image	Image (toujours non-saisissable)	<IMG SRC="..." ...>
Sous-table	Pas de support HTML	Aucune
BLOB	Pas de support HTML (****)	Aucune

(*) ou Texte seul si non-saisissable

(**) si la valeur de type Texte se compose de plusieurs lignes

(***) si la valeur de type Texte ne se compose que d'une ligne ou est vide

(****) la commande ENVOYER BLOB HTML vous permet toutefois d'envoyer tout type de champ ou variable BLOB aux browsers.

Note : Les variables saisissables se comportent comme les champs du même type.

Objets de formulaires

Lorsqu'un formulaire 4D est converti en page HTML, les objets de formulaire sont traduits de la manière suivante :

Objet 4D	Objet HTML équivalent	Balise HTML
Ligne	Ligne horizontale (1)	<HR>
Rectangle	Rectangle	Géré par les CSS1
Ovale	Pas de support HTML	Aucune
Rectangle arrondi	Pas de support HTML	Aucune
Image statique	Image ou Image interactive (2)	<IMG SRC="..."> <INPUT Type="image" ...>
Zone de groupe	Texte	Texte avec balises si nécessaire
Texte statique	Texte	Texte avec balises si nécessaire
Bouton	Bouton "submit"	<INPUT Type="submit" ...>
Bouton par défaut	Bouton "submit"	<INPUT Type="submit" ...>
Bouton radio	Bouton radio (3)	<INPUT Type="radio" ...>

Case à cocher	Case à cocher	<INPUT Type="checkbox" ...>
Pop up/Liste déroulante	Liste déroulante	<SELECT ...>...</SELECT>
Combo Box	Liste déroulante	<SELECT ...>...</SELECT>
Zone de défilement	Zone de défilement (4)	<SELECT ...>...</SELECT>
Onglet	Liste d'URLs (5)	<A HREF="/4DTAB/ 4DVar.Onglet...">
Bouton invisible	Référez-vous à la note 2	
Bouton inversé	Référez-vous à la note 2	
Bouton 3D	Référez-vous à la note 2	
Grille de boutons	Référez-vous à la note 2	
Graphe	Image (non-saisissable)	<IMG SRC="..." ...>
Plug-in	Texte, Image ou Image interactive (6)	Texte avec balises si nécessaire <IMG SRC="..." ...> <INPUT Type="image" ...>

Les objets suivants ne sont pas encore convertis en HTML et sont donc ignorés :

Liste hiérarchique, Menu déroulant hiérarchique, Sous-formulaire, Bouton radio image, Thermomètre, Règle, Cadran, Menu image, Bouton image, Case à cocher 3D, Bouton radio 3D, Séparateur.

Notes

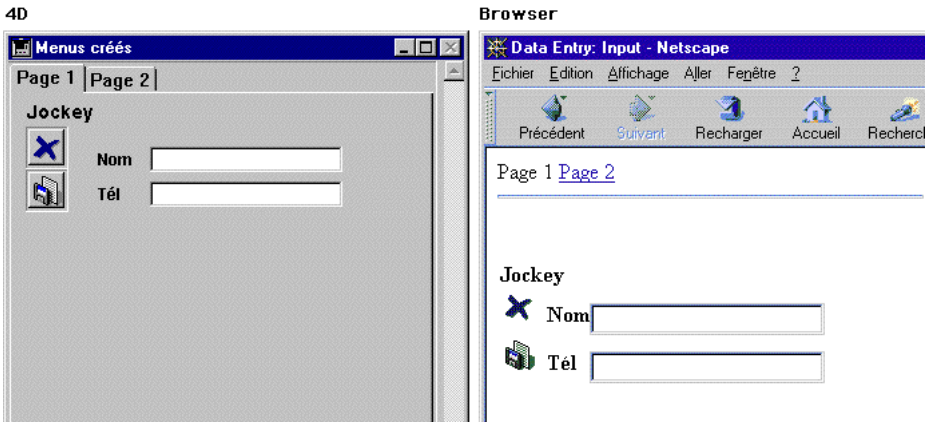
1. Les lignes non horizontales ne sont pas supportées par le HTML ; elles sont donc ignorées.

2. Les boutons non-visibles sont des objets de type Bouton invisible, Bouton inversé, Bouton 3D et Grille de boutons. Lorsqu'une image statique ne chevauche aucun bouton non-visible, elle est traduite comme une image statique. Si elle chevauche au moins un bouton non-visible, elle est traduite comme une "image interactive" gérée sur le serveur (Server-side Image Map).
Sur le browser Web, l'image est traitée comme une *Server-side Image Map*. Du côté de 4D, lorsque le formulaire est reçu, 4D recalcule la position du clic pour générer un événement Sur clic souris pour le bouton correspondant, comme si l'utilisateur avait réellement cliqué sur le bouton. Gérer des boutons non-visibles est donc très simple, dans la mesure où ils recouvrent ou sont recouverts par des images statiques. Vous contrôlez ces boutons dans la méthode formulaire ou dans leurs méthodes objets, tout comme vous le faites pour l'interface 4D standard. C'est aussi un moyen simple de gérer l'interactivité des images Web (Web Image Mapping). Si un bouton non-visible ne chevauche aucune image statique, il est ignoré lors de la conversion.

3. Le regroupement des boutons radio est maintenu lors de la conversion.

4. Les zones de défilement groupées ne sont pas supportées par le HTML. 4D les traduit en tant que zones de défilement indépendantes situées sur la même ligne.

5. Les onglets (de type tableau ou créés à l'aide de valeurs par défaut définies dans les Propriétés de l'objet) sont convertis sous forme de listes d'URL :



Si les valeurs sont des chaînes vides, 4D affiche 1, 2, 3... sur le browser.

6. Les zones de plug-ins sont publiables sur le Web, étant au préalable converties en HTML, en Image ou en "Image Map". Cette dernière solution permet de gérer les clics souris à l'intérieur de la zone de plug-in (par exemple, la zone _AP External clock de 4D_Pack est publiée en Image, le plug-in intégré 4D Chart est publié en Image Map). La manière dont une zone de plug-in incluse dans un formulaire est publiée sur le Web dépend des spécifications de l'éditeur du plug-in.

Visualiser sélection / Modifier sélection

• Le mécanisme UserSet n'est pas supporté.
• Un mécanisme de sélection automatique de pages est fourni par 4D. Pour plus d'informations, reportez-vous à la description de la commande FIXER LIMITES AFFICHAGE WEB.

Commandes 4D

Lorsque vous développez une base 4D pour le Web, vous pouvez vous demander ce qui va se produire lorsque telle ou telle commande sera appelée. La commande produira-t-elle un effet sur la machine du serveur Web ou sur celle du browser Web ? Le process de connexion Web s'exécute sur la machine du serveur Web, mais l'interface utilisateur est affichée à distance sur le browser Web connecté. Par conséquent, pour le développement d'une base Web, les commandes 4D peuvent être classées de la manière suivante :

Les commandes qui ne sont pas affectées par leur exécution dans un process de connexion Web

Une commande telle que CREER ENREGISTREMENT fonctionne dans le process courant ; dans ce cas, elle crée un enregistrement depuis le process de connexion Web. Le même principe est valable pour les commandes telles que Largeur ecran, qui retourne la largeur de l'écran de la machine du serveur Web (la machine sur laquelle le process est exécuté).

Les commandes qui comportent des fonctionnalités intégrées supplémentaires pour le support transparent du Web

Nom de la commande	Commentaires
AJOUTER ENREGISTREMENT	Conversion automatique du formulaire, gestion des formulaires multi-pages
ALERTE	Conversion automatique de la boîte de dialogue
CONFIRMER	Conversion automatique de la boîte de dialogue
DIALOGUE	Conversion automatique du formulaire, gestion des formulaires multi-pages
VISUALISER SELECTION	Conversion automatique du formulaire Mécanisme intégré de pagination de la sélection Le mécanisme UserSet n'est pas géré
MODIFIER ENREGISTREMENT	Conversion automatique du formulaire, gestion des formulaires multi-pages
MODIFIER SELECTION	Conversion automatique du formulaire Mécanisme intégré de pagination de la sélection Le mécanisme UserSet n'est pas géré
CHERCHER	Boîte de dialogue standard de recherche gérée
CHERCHER PAR EXEMPLE	Conversion automatique du formulaire, gestion des formulaires multi-pages
Demander	Conversion automatique de la boîte de dialogue
FIXER MINUTEUR	Support de l'événement formulaire Sur minuteur
REDESSINER	Mise à jour du formulaire affiché sur le browser

ATTENTION : En mode sans contexte, l'appel d'une commande 4D affichant un élément d'interface (ALERTE, DIALOGUE...) entraîne l'arrêt du traitement (cf. section Services Web, Mode sans contexte).

Les commandes à utiliser avec précaution

Les commandes suivantes s'exécutent localement sur la machine du serveur Web.

Par exemple, vous pouvez demander l'impression d'une sélection à partir du browser Web. Cependant, l'impression sera effectuée sur la machine du serveur Web.

De plus, lorsqu'un élément d'interface est appelé, il s'affiche sur la machine du serveur Web, par exemple Ouvrir document(""), par opposition à Ouvrir document("Ce document"). Vous devrez éviter ce genre d'appel car le browser Web attendra une réponse jusqu'à ce que la boîte de dialogue soit refermée sur la machine du serveur Web. En revanche, il est tout à fait possible d'appeler ces routines lorsqu'aucune boîte de dialogue n'est requise.

Nom de la commande	Commentaires
Ajouter a document	OK, si aucune boîte de dialogue de fichier n'est appelée
BEEP	Emet un bip sur la machine du serveur Web
Creer document	OK, si aucune boîte de dialogue de fichier n'est appelée
AFFICHER ENREGISTREMENT	Ne fait rien
ECRITURE DIF	OK, si aucune boîte de dialogue de fichier n'est appelée
ECRITURE SYLK	OK, si aucune boîte de dialogue de fichier n'est appelée
ECRITURE ASCII	OK, si aucune boîte de dialogue de fichier n'est appelée
LECTURE DIF	OK, si aucune boîte de dialogue de fichier n'est appelée
LECTURE SYLK	OK, si aucune boîte de dialogue de fichier n'est appelée
LECTURE ASCII	OK, si aucune boîte de dialogue de fichier n'est appelée
CHARGER ENSEMBLE	OK, si aucune boîte de dialogue de fichier n'est appelée
LIRE VARIABLES	OK, si aucune boîte de dialogue de fichier n'est appelée
MESSAGE	Les messages s'affichent sur la machine du serveur Web
Ouvrir document	OK, si aucune boîte de dialogue de fichier n'est appelée
Creer fenetre externe	La fenêtre s'ouvre sur la machine du serveur Web
Ouvrir fichier ressources	OK, si aucune boîte de dialogue de fichier n'est appelée
Creer fenetre	La fenêtre s'ouvre sur la machine du serveur Web
JOUER SON	Le son est joué sur la machine du serveur Web
IMPRIMER LIGNE	OK, si aucune boîte de dialogue d'impression n'est appelée
IMPRIMER ETIQUETTES	OK, si aucune boîte de dialogue d'impression n'est appelée
IMPRIMER ENREGISTREMENT	OK, si aucune boîte de dialogue d'impression n'est appelée
IMPRIMER SELECTION	OK, si aucune boîte de dialogue d'impression n'est appelée
QUITTER 4D	Géré, vous pouvez quitter le serveur Web à distance
STOCKER ENSEMBLE	OK, si aucune boîte de dialogue de fichier n'est appelée
ECRIRE VARIABLES	OK, si aucune boîte de dialogue de fichier n'est appelée
FIXER HISTORIQUE	OK, si aucune boîte de dialogue de fichier n'est appelée
REGLER SERIE	OK, si aucune boîte de dialogue de fichier n'est appelée (documents)
TRACE	La fenêtre de débogage s'affiche sur la machine du serveur Web

Les commandes qui ne sont pas supportées par les process de connexion Web

Nom de la commande	Commentaires
AJOUTER SEGMENT DE DONNEES	N'appelez PAS cette commande dans un process de connexion Web. Cette commande n'a pas encore été adaptée pour le Web.
AJOUTER SOUS ENREGISTREMENT	N'appelez PAS cette commande dans un process de connexion Web. Cette commande n'a pas encore été adaptée pour le Web.
CHANGER UTILISATEUR	N'appelez PAS cette commande dans un process de connexion Web. Cette commande n'a pas encore été adaptée pour le Web.
CHANGER PRIVILEGES	N'appelez PAS cette commande dans un process de connexion Web. La fenêtre des mots de passe s'affiche sur la machine de 4D. Le browser attendra jusqu'à ce que la fenêtre soit fermée.
GRAPHE SUR SELECTION	N'appelez PAS cette commande dans un process de connexion Web. Cette commande n'a pas encore été adaptée pour le Web.
MODIFIER SOUS ENREGISTREMENT	N'appelez PAS cette commande dans un process de connexion Web. Cette commande n'a pas encore été adaptée pour le Web.
TRIER	Gestion sous forme programmée seulement La boîte de dialogue de Tri n'est pas encore adaptée au Web.
PARAMETRES IMPRESSION	N'appelez PAS cette commande dans un process de connexion Web. Les boîtes de dialogue d'impression s'affichent sur la machine de 4D. Le browser attendra jusqu'à ce que les boîtes de dialogue soient fermées.
ETAT	N'appelez PAS cette commande dans un process de connexion Web. La fenêtre de l'éditeur d'Etats semi-automatiques s'affiche sur la machine de 4D. Le browser attendra jusqu'à ce que la fenêtre soit fermée.

WML

Le serveur Web 4D supporte la technologie WML (*Wireless Markup Language*). Cette fonctionnalité autorise la consultation et la saisie d'informations dans des bases 4D à l'aide d'un téléphone mobile ou d'un assistant électronique personnel (PDA).

Note : Le langage WML, associé au protocole WAP (*Wireless Application Protocol*), est développé conjointement par plusieurs sociétés. Le WAP propose un ensemble d'outils de communication réseau, destiné à permettre aux téléphones mobiles et aux PDA de visualiser du texte publié dans des pages Web. La technologie WML est ouverte et libre de droits. Pour plus d'informations sur les spécifications du WML, veuillez consulter le site Web de Phone.com : http://www.phone.com/

La visualisation et la saisie d'informations s'effectuent par l'intermédiaire de pages WML utilisant le mécanisme des balises 4DVAR ou 4DSCRIPT.

Voici la liste des documents WML acceptés par le serveur Web 4D

Extension	Type MIME	Description
.wml	text/vnd.wap.wml	Pages WML (toujours analysées par 4D*)
.wmls	text/vnd.wap.wmlscript	Scripts WML (côté client)
.wmlc	application/vnd.wap.wmlc	Pages WML binaires
.wmlsc	application/vnd.wap.wmlscript	Scripts WML binaires
.wbmp	image/vnd.wap.wbmp	Images bitmap destinées aux mobiles (parfois non supportées)

* Permet l'insertion dynamique de données via les balises 4DVAR et 4DSCRIPT.

XML

Le serveur Web 4D accepte les documents .xml, .xsl et .dtd. Ces documents sont respectivement envoyés avec le type MIME "text/xml", "text/xsl" et "text/xml".

Quel que soit le mode (contextuel ou sans contexte) depuis lequel ces documents sont envoyés, 4D analyse leur contenu et traite les éventuelles balises 4DVAR ou 4DSCRIPT qu'ils contiennent, afin de générer du XML dynamique.

Note : Il n'est pas possible d'envoyer du XML depuis l'intérieur même d'un formulaire 4D, en utilisant une balise du type {mapage.xml} insérée dans un texte statique.

Référence

Encapsulation HTML et Javascript, Services Web, Support du HTML.

Note préliminaire

Nous vous conseillons de lire les sections suivantes avant de travailler avec cette partie de la documentation :

- Services Web, Présentation
- Services Web, Configuration
- Services Web, Premiers pas (Partie I)
- Services Web, Premiers pas (Partie II)
- Services Web, Process de connexion Web
- Services Web, Support du HTML

Il y a plusieurs façons d'encapsuler du code HTML dans votre application 4D :

1. Avec les commandes ENVOYER FICHIER HTML ou ENVOYER BLOB HTML, vous pouvez envoyer une page Web stockée sur disque.

2. Avec la commande ENVOYER FICHIER HTML, vous pouvez envoyer directement du code HTML aux navigateurs Web.

3. Un objet texte statique de formulaire 4D, comme "{page.HTM}", insère le document HTML "page.HTM" dans le formulaire 4D à l'endroit où se trouve l'objet texte.

Notes :

- En mode sans contexte, il n'est pas possible d'utiliser cette fonctionnalité. Pour plus d'informations, reportez-vous à la section Services Web, Mode sans contexte.
- Dans certains cas, la conversion HTML de formulaires 4D créés en version 6.0.x contenant une référence à un document HTML ({mapage.htm}) peut ne pas donner le résultat escompté avec 4D 6.7. Dans ce cas, il est possible de modifier le mode de conversion des formulaires à l'aide de la commande FIXER PARAMETRE BASE.

4. Toute variable texte 4D peut encapsuler du code HTML dans un formulaire 4D, si son premier caractère a le code ASCII 1 (par exemple, vtHTML:=Caractere(1)+"...Code HTML...").

Dans les deux derniers cas, le formulaire qui en résulte, côté browser Web, est la combinaison d'objets HTML et d'objets 4D. Notez qu'en utilisant un **objet de texte statique**, vous insérez un document dans son intégralité (en fait, tout ce qui est compris entre les balises <BODY> et </BODY>. En revanche, si vous utilisez une **variable texte**, vous insérez des morceaux de code.

Avec ENVOYER FICHIER HTML, ENVOYER BLOB HTML ou un objet texte statique de formulaire, vous pouvez soit utiliser un document HTML existant, soit construire par le langage un document que vous sauvegardez sur disque et auquel vous vous référez par la suite. Avec une variable texte située dans le formulaire, vous pouvez construire le code HTML en mémoire.

Associer des objets HTML à des méthodes 4D

Quelle que soit la manière dont vous encapsulez le HTML dans votre application 4D, vous pouvez associer un objet HTML à une méthode 4D en créant pour cet objet un lien. L'URL doit être de la forme :
• /4DMETHOD/Nom_Méthode pour le mode contextuel,
• /4DACTION/Nom_Méthode pour le mode sans contexte,
où Nom_Méthode est le nom de la méthode projet 4D que vous voulez exécuter lorsque l'objet HTML est activé. Des exemples de méthodes 4D liées à des objets HTML en mode contextuel sont fournis dans la section Services Web, Premiers pas (Partie II). Des exemples de méthodes 4D liées à des objets HTML en mode sans contexte sont fournis dans la section Services Web, Mode sans contexte.

Important
• Si vous envoyez un fichier HTML, vous pouvez associer des méthodes 4D avec n'importe quel type d'objet HTML pouvant être lié. Cependant, rappelez-vous que si vous voulez qu'une méthode 4D avec l'action POST exécute un ENVOYER FICHIER HTML("") pour stopper le mode HTML, votre page HTML doit comporter un bouton Submit pour que l'action POST de la page puisse être exécutée (la page étant soumise au Serveur Web 4D).
• En revanche, si vous encapsulez du code HTML dans un formulaire 4D (texte statique ou variable texte), vous ne pouvez pas utiliser de boutons Submit. Vous pouvez avoir des liens renvoyant à des méthodes 4D, mais vous ne pouvez pas spécifier l'action POST car c'est 4D qui construit la page, et non vous.

Associer des objets HTML à des variables 4D - 4D vers Navigateur

Vous pouvez associer des objets HTML avec des variables 4D.

Notes :
• Vous devez utiliser des variables process.
• Comme le HTML est orienté Texte, vous travaillerez généralement avec des variables Texte. Toutefois, vous pouvez également utiliser des variables BLOB (ce qui vous permet de vous affranchir de la limite des 32 000 caractères inhérente aux variables de type texte). Il vous suffit pour cela de générer des BLOBs en mode Texte sans longueur.

Tout d'abord, un objet HTML peut voir sa valeur initialisée par la valeur d'une variable 4D.

Ensuite, une fois qu'une page Web est envoyée en retour, la valeur d'un objet HTML peut être retournée dans une variable 4D. Pour cela, dans le code de la page HTML, vous créez un objet HTML **dont le nom est le même que celui de la variable process 4D** que vous voulez associer. Ce point est étudié plus loin dans le paragraphe “Associer des objets HTML avec des variables 4D - Navigateur vers 4D ”, après la section sur JavaScript.

Note : En mode sans contexte, il n’est pas possible faire référence à des variables images 4D. Pour plus d'informations, reportez-vous à la section Services Web, Mode sans contexte.

Revenons au premier point. Vous pouvez donner par programmation des valeurs par défaut aux objets HTML en incluant <!--4DVAR NomVar--> dans, par exemple, le champ **valeur** de l'objet HTML ; NomVar est le nom de la variable process 4D telle qu'elle est définie dans le Process de connexion Web — nom que vous mettez entre <!-- -->, c'est-à-dire la notation standard pour les commentaires HTML.

Note : Certains éditeurs HTML n'acceptent pas la saisie de la valeur <!--4DVAR NomVar--> dans le champ **valeur** des objets HTML. Dans ce cas, vous devrez la placer directement dans le code HTML.

Note de compatibilité : Dans la version 6.0.x de 4D, l'insertion de variables 4D dans les pages statiques s'effectuait à l'aide d'une notation utilisant des crochets ([NomVar]). Si vous travaillez avec une base créée avec une version précédente de 4D et souhaitez utiliser la notation standard <!—4DVAR NomVar—>, assurez-vous que l'option "Utiliser les commentaires 4DVAR au lieu des crochets" dans la boîte de dialogue des Propriétés de la base est bien cochée (reportez-vous à la section Services Web, Paramétrages du serveur Web).

En fait, la syntaxe <!—4DVAR NomVar—> permet d'insérer des données 4D partout dans la page HTML. Si, par exemple, vous écrivez :

<P>Bienvenue dans <!--4DVAR vtSiteName-->!</P>

La valeur de la variable 4D vtSiteName sera insérée dans la page HTML.

Examinons un exemple :

```
    ` Voici le code 4D assignant "4D4D" à la variable process vs4D
vs4D:="4D4D"
    ` Puis envoyer la page HTML "AnyPage.HTM"
ENVOYER FICHIER HTML("AnyPage.HTM")
```

Le source de la page HTML AnyPage.HTM est le suivant :

```
<HTML>
<HEAD>
   <TITLE>AnyPage</TITLE>
   <SCRIPT LANGUAGE="JavaScript"><!--

       function Is4DWebServer(){
          return (document.frm.vs4D.value=="4D4D")
       }

       function HandleButton(){
          if (Is4DWebServer()){
             alert("You are connected to 4D Web Server!")
          } else {
             alert("You are NOT connected to 4D Web Server!")
          }
       }


   //--></SCRIPT>
</HEAD>
<BODY>
<FORM action="/4DMETHOD/WWW_STD_FORM_POST" method="POST" name="frm">

<P><INPUT TYPE="hidden" NAME="vs4D" VALUE="<!--4DVAR vs4D-->"</P>

<P><A HREF="JavaScript:HandleButton()"><IMG SRC="AnyGIF.GIF" BORDER=0
ALIGN=bottom></A></P>

<P><INPUT TYPE="submit" NAME="bOK" VALUE="OK"></P>

</FORM>
</BODY>
</HTML>
```

En mode contextuel, avant l'envoi d'une page HTML (document HTML ou formulaire 4D traduit), 4D scrute toujours le source HTML pour rechercher les objets renvoyant à des variables 4D (et recalculer les URLs des liens, comme nous le verrons ensuite).
En mode sans contexte, l'analyse du source HTML par 4D est effectuée sous certaines conditions (reportez-vous à la section Services Web, Mode sans contexte).

Dans le code HTML ci-dessus, vous remarquez le type de saisie **hidden** dont le nom est vs4D. La valeur de cet objet est mise à la valeur texte "<!--4DVAR vs4D-->". Comme la méthode projet qui envoie le fichier HTML a préalablement défini la variable process vs4D, 4D remplace la valeur de l'objet HTML et la met à "4D4D", la valeur de la variable 4D.

Vous notez aussi la fonction encapsulée JavaScript Is4DWebServer qui teste la valeur de l'objet HTML vs4D. Voici l'astuce : si la page HTML est envoyée par 4D, la valeur de l'objet est changée en "4D4D" ; en revanche, si la page HTML est envoyée par une autre application (par exemple 4D WebStar sur Macintosh), l'objet conserve sa valeur telle qu'elle est définie dans la page, c'est-à-dire "[vs4D]". En testant la valeur de cet objet, par JavaScript, vous pouvez détecter à l'intérieur de la page, côté Browser Web, si cette page est ou non envoyée par 4D.

Ce premier exemple montre comment vous pouvez construire des pages HTML “intelligentes”, qui comportent des caractéristiques supplémentaires lorsqu'elles sont envoyées par 4D tout en restant compatibles avec d'autres serveurs de Web.

Important : Rappelons que vous ne devez associer que des variables process. De plus, vous ne pouvez pas, dans la version actuelle du programme, associer de tableau 4D à un objet HTML SELECT. En revanche, chaque élément d'un objet SELECT peut se référer à des variables 4D séparées (par exemple, le premier élément à V1, le second à V2, et ainsi de suite).

Une association dans le sens 4D vers navigateur Web fonctionne avec n'importe quel procédé d'encapsulation (ENVOYER FICHIER HTML, ENVOYER BLOB HTML, texte statique ou variable texte ou BLOB dans un formulaire 4D).

Insérer du code HTML dans des variables 4D

Vous pouvez insérer du code HTML dans des variables 4D incluses dans des pages statiques. Lorsque la page est affichée sur le navigateur, la valeur de la variable est remplacée par le code HTML, qui est alors interprété par le navigateur.

Pour insérer du code HTML dans des variables 4D, vous disposez de deux possibilités :
• Faire débuter la variable 4D par le code ASCII 1 (par exemple vtHTML:=Caractere(1)+"...Code HTML...") et l'insérer dans la page HTML via la balise <!--4DVAR vtHTML-->.
• Insérer une variable 4D standard (par exemple vtHTML:="...Code HTML...") dans la page HTML via la balise <!--4DHTMLVAR vtHTML-->.
Vous pouvez utiliser une variable de type Texte ou BLOB (généré en mode Texte sans longueur).

Pour plus d'informations, reportez-vous à la section “Services Web, Balises HTML 4D”.

Encapsulation du JavaScript

Le code source JavaScript encapsulé dans les documents HTML est supporté, ainsi que l'insertion de fichiers JavaScript .js dans des documents HTML (par exemple <SCRIPT SRC="...").

Avec ENVOYER FICHIER HTML ou ENVOYER BLOB HTML, vous envoyez une page que vous avez préparée avec un éditeur HTML ou construite avec 4D et sauvegardée comme document sur disque. Dans les deux cas, vous avez tout contrôle sur la page et, par exemple, vous pouvez insérer des scripts JavaScript dans la section HEAD du document et utiliser des scripts avec une balise FORM. Ainsi, dans l'exemple ci-dessus, le script renvoie le formulaire "frm" car vous avez donné un nom au formulaire. Vous pouvez également déclencher, accepter ou rejeter la soumission du formulaire au niveau de la balise FORM.

Si vous encapsulez de l'HTML dans un formulaire 4D, vous n'avez pas de contrôle sur la section HEAD (en-tête) ni sur la déclaration du FORM (formulaire). L'aire d'action des scripts est donc différente. Ainsi, vous ne pouvez pas accéder au formulaire HTML par son nom. Cependant, comparez la fonction JavaScript Is4DWebServer de l'exemple ci-dessus avec celui-ci :

```
function Is4DWebServer(){
   return (document.forms[0].vs4D.value=="4D4D")
}
```

Les deux fonctions font la même chose. La seconde utilise la propriété forms de l'objet HTML document pour accéder à l'objet par l'élément forms[0]. Le résultat est que tout fonctionne, même si vous ne connaissez pas le nom du formulaire que 4D a ou n'a pas donné à la page HTML traduite.

Note : 4D supporte le transport d'Applets Java.

Associer des objets HTML à des variables 4D - Navigateur vers 4D

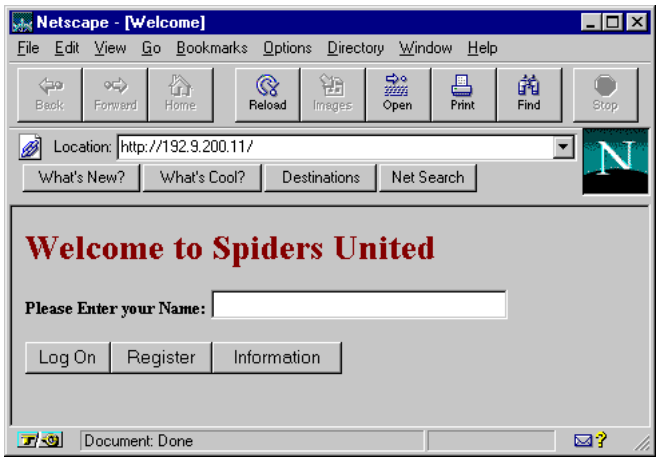
Quand vous envoyez une page HTML avec ENVOYER FICHIER HTML ou ENVOYER BLOB HTML, vous pouvez aussi associer des variables 4D avec des objets HTML dans le sens "navigateur Web vers 4D". L'association fonctionne dans les deux sens : lorsque la page HTML est retournée, 4D recopie les valeurs des objets HTML dans des variables process 4D.

Attention : La récupération des valeurs dans les variables process 4D n'est possible qu'avec des pages HTML envoyées avec ENVOYER FICHIER HTML ou ENVOYER BLOB HTML. Dans le cas de HTML encapsulé dans un formulaire 4D, la récupération des valeurs est limitée aux objets 4D effectivement placés dans ce formulaire.

Considérons cette page de code HTML :

```
<HTML>
<HEAD>
   <TITLE>Welcome</TITLE>
   <SCRIPT LANGUAGE="JavaScript"><!--

         function GetBrowserInformation(formObj){
            formObj.vtNav_appName.value = navigator.appName
            formObj.vtNav_appVersion.value = navigator.appVersion
            formObj.vtNav_appCodeName.value = navigator.appCodeName
            formObj.vtNav_userAgent.value = navigator.userAgent
            return true
         }

         function LogOn(formObj){
            if(formObj.vtUserName.value!=""){
               return true
            } else {
               alert("Enter your name, then try again.")
               return false
            }
         }
    //--></SCRIPT>
</HEAD>
<BODY>
<FORM action="/4DMETHOD/WWW_STD_FORM_POST" method="POST" name="frmWelcome"
                        onsubmit="return GetBrowserInformation(frmWelcome)">

<H1>Welcome to Spiders United</H1>

<P><B>Please Enter your Name:</B>
   <INPUT TYPE="text" NAME="vtUserName" VALUE="<!--4DVAR vtUserName-->" SIZE=30></P>
<P>
   <INPUT TYPE="submit" NAME="vsbLogOn" VALUE="Log On" onclick="return LogOn(frmWelcome)">
   <INPUT TYPE="submit" NAME="vsbRegister" VALUE="Register">
   <INPUT TYPE="submit" NAME="vsbInformation" VALUE="Information">
</P>

<P>
   <INPUT TYPE="hidden" NAME="vtNav_appName" VALUE="">
   <INPUT TYPE="hidden" NAME="vtNav_appVersion" VALUE="">
   <INPUT TYPE="hidden" NAME="vtNav_appCodeName" VALUE="">
   <INPUT TYPE="hidden" NAME="vtNav_userAgent" VALUE="">
</P>
</FORM>
</BODY>
</HTML>
```

Lorsque la page est envoyée au navigateur Web par 4D, elle se présente ainsi :



Voici les caractéristiques de cette page :

• Elle comporte trois boutons Submit : vsbLogOn, vsbRegister et vsbInformation.
• Le Submit du formulaire quand vous cliquez sur LogOn est d'abord traité par la fonction JavaScript LogOn. Si aucun nom n'a été saisi, le formulaire n'est pas envoyé à 4D et une alerte JavaScript est affichée.
• Le formulaire comporte une méthode 4D POST et un script Submit (GetBrowserInformation) qui copie les propriétés du Navigator dans quatre objets cachés dont les noms commencent par vtNav_App.
• Vous remarquez aussi l'objet vtUserName dont la valeur initiale est <!--4DVAR vtUserName-->.

Examinons maintenant la méthode 4D (nommé ici WWW Welcome) qui envoie cette page HTML à l'aide de la commande ENVOYER FICHIER HTML. Cette méthode est appelée par la Méthode base Sur connexion Web.

```
` Méthode projet WWW Welcome
` WWW Welcome -> Booleén
` WWW Welcome -> Oui = La session peut commencer

C_BOOLEEN($0)
$0:=Faux

` Objets de saisie HTML cachés retournant l'information sur le Browser
C_TEXTE(vtNav_appName;vtNav_appVersion;vtNav_appCodeName;vtNav_userAgent)
vtNav_appName:=""
vtNav_appVersion:=""
vtNav_appCodeName:=""
vtNav_userAgent:=""
```

```
   ` Objet texte saisie HTML où le nom de l'utilisateur est saisi
C_TEXTE(vtUserName)
vtUserName:=""

   ` Valeurs du HTML bouton Submit
C_ALPHA(31;vsbLogOn;vsbRegister;vsbInformation)

Repeter
      ` Ne pas oublier d'initialiser les valeurs des boutons Submit!
   vsbLogOn:=""
   vsbRegister:=""
   vsbInformation:=""
      ` Envoyer la page Web
   ENVOYER FICHIER HTML("Welcome.HTM")
      ` Tester les valeurs des boutons Submit pour détecter si l'un d'entre eux
      ` a reçu un clic souris
   Au cas ou

         ` Le bouton Log On bouton a été cliqué
      : (vsbLogOn # "")
         CHERCHER([WWW Users];[WWW Users]User Name=vtUserName)
         $0:=(Enregistrements trouves([WWW Users])>0)
         Si ($0)
            WWW POST EVENT ("Log On";WWW Log information )
               ` La méthode WWW POST EVENT sauvegarde l'information
               ` dans une table de la base
         Sinon
            CONFIRMER("Ce nom d'utilisateur est inconnu. Voulez-vous l'enregistrer?")
            $0:=(OK=1)
            Si ($0)
               $0:=WWW Register
                  ` La méthode WWW Register permet à un nouvel utilisateur Web de
                  ` s'enregistrer
            Fin de si
         Fin de si

         ` Le bouton Register a reçu un clic souris
      : (vsbRegister # "")
         $0:=WWW Register

            ` Le bouton Information a reçu un clic souris
      : (vsbInformation # "")
         DIALOGUE([User Interface];"WWW Information")
   Fin de cas
Jusque (Non(◊vbWebServicesOn) | $0)
```

Note : Vous pouvez également obtenir le nom et la valeur de chaque variable contenue dans un formulaire Web “soumis” (c.-à-d. envoyée au serveur Web) à l'aide de la commande LIRE VARIABLES FORMULAIRE WEB.

Voici les caractéristiques de cette méthode :

• Les variables 4D vtNav_appName, vtNav_appVersion, vtNav_appCodeName et vtNav_userAgent (liées aux objets HTML de même nom) reçoivent les valeurs assignées aux objets HTML par le script JavaScript GetBrowserInformation. C'est simple : initialisez les variables en chaînes de caractères, et ensuite récupérez les valeurs après que la page ait été soumise.

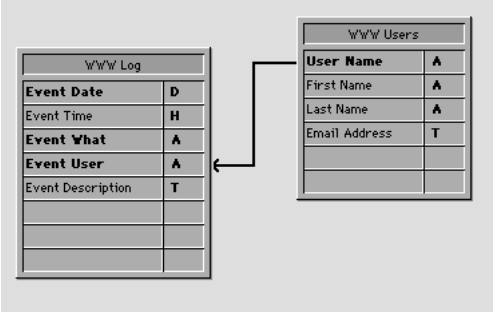
• Les variables 4D vsbLogOn, vsbRegister et vsbInformation sont liées aux trois boutons Submit. Notez que ces variables sont remises à leur valeur vide chaque fois que la page est envoyée au browser. Quand le submit est effectué par l'un de ces trois boutons, le browser retourne à 4D la valeur du bouton sur lequel on a cliqué. Comme les variables 4D sont initialisées à chaque fois, la variable qui n'est plus égale à chaîne vide vous indique sur quel bouton on a cliqué. Les deux autres variables sont des chaînes vides, non pas parce que le browser a retourné des chaînes vides, mais parce qu'il “n'a rien dit” à leur sujet et, en conséquence, 4D les a laissées inchangées. C'est la raison pour laquelle il est nécessaire de réinitialiser ces variables toutes les fois que la page est envoyée au browser.

C'est **de cette manière que vous pouvez savoir** quel bouton Submit a été activé quand plusieurs boutons Submit se trouvent sur la page Web. Notez que les boutons 4D (dans un formulaire 4D) sont des variables numériques. En revanche, en HTML, tous les objets sont des objets texte.

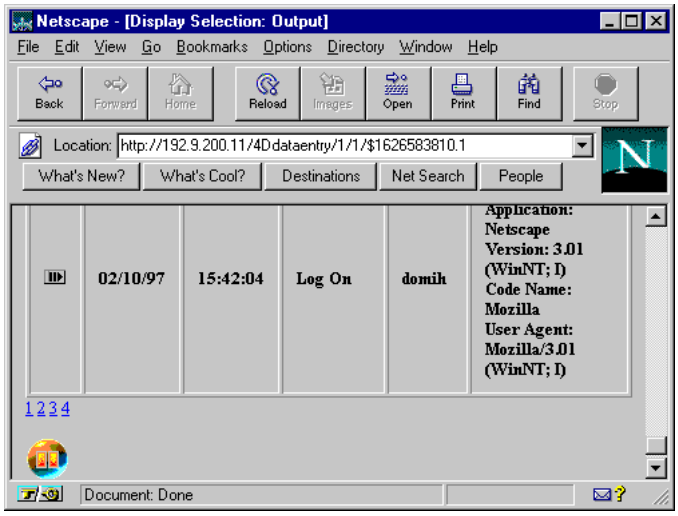
Si vous liez une variable 4D avec un objet SELECT, vous liez aussi une variable texte. Dans 4D, pour tester quel élément d'une liste déroulante a été sélectionné, vous testez la valeur numérique du tableau 4D. En HTML, c'est la valeur de l'élément sélectionné qui est retournée dans la variable 4D liée à l'objet HTML.

Quel que soit l'objet que vous associez à une variable 4D, la valeur retournée est de type Texte. Donc, vous devez associer des variables 4D process de type Alpha ou Texte.

Un point intéressant de cet exemple (partiel) est qu'une fois que vous avez obtenu l'information sur le browser, vous pouvez stocker ces valeurs dans une table 4D, et là encore vous associez le Web et les capacités de base de données de 4D. C'est ce que réalise la méthode projet WWW POST EVENT. Elle ne “poste” pas un événement, elle sauvegarde les informations de la session Web dans des tables du type de celles-ci :



Une fois que vous avez sauvegardé ces informations dans une table, vous pouvez, avec d'autres méthodes projet, renvoyer l'information à l'utilisateur Web. Pour cela, vous utilisez simplement un CHERCHER pour trouver la bonne information et un VISUALISER SELECTION pour montrer les enregistrements [WWW Log]. L'image ci-dessous présente les informations de connexion que l'utilisateur Web peut obtenir s'il est un utilisateur enregistré du site Web :



Compte tenu des possibilités des liens 4D/HTML dans cet exemple et des informations que vous pouvez demander aux utilisateurs par des dialogues HTML ou des formulaires 4D, vous pouvez concevoir des capacités d'administration très intéressantes pour votre site Web.

Associer des objets HTML à des variables 4D - "Image Mapping"

Comme cela est décrit dans la section Services Web, Support du HTML, quand un formulaire 4D est utilisé comme page Web, 4D peut fournir une image interactive côté serveur ("Server-side Image Mapping") au moyen de boutons invisibles placés par-dessus une image statique.

Si vous envoyez un document HTML avec ENVOYER FICHIER HTML ou ENVOYER BLOB HTML, vous pouvez aussi lier une variable 4D avec des objets HTML Image interactive (INPUT TYPE="IMAGE"). Par exemple, vous créez un objet HTML Image interactive dont le nom est bImageMap (vous prenez le nom que vous voulez). Chaque fois que vous cliquez sur l'image côté browser, un submit avec la position du clic est renvoyé au Serveur Web 4D. Pour récupérer les coordonnées de ce clic (exprimé par rapport au coin supérieur gauche de l'image), il vous suffit simplement de lire la valeur des variables process 4D bImageMap_X et bImageMap_Y (de type Entier long) qui contiennent les coordonnées horizontales et verticales du clic. Ces variables doivent impérativement être initialisées dans la méthode projet *COMPILER_WEB*.

Dans la page HTML, vous écrivez, par exemple :

```
<P><INPUT TYPE="image" SRC="MonImage.GIF" NAME="bImageMap" BORDER=0></P>
```

La méthode 4D qui envoie la page HTML contient simplement :

```
ENVOYER FICHIER HTML("CettePage.HTM")
```

Dans la méthode projet *COMPILER_WEB*, vous écrivez :

```
C_ENTIER LONG(bImageMap_X;bImageMap_Y)
bImageMap_X:=-1    `Initialisation de la variable
bImageMap_Y:=-1    `Initialisation de la variable
```

Note : Pour plus d'informations sur la méthode *COMPILER_WEB*, reportez-vous à la section “Services Web, URLs et actions de formulaires”.

Ensuite, dans la méthode 4D appelée par l'action POST du formulaire — ou dans la méthode courante une fois que la méthode d'action POST a émis un ENVOYER FICHIER HTML ("") — vous récupérez les coordonnées du clic dans les variables bImageMap_X et bImageMap_Y :

```
Si ((bImageMap_X#-1)&(bImageMap_Y#-1))
   ` Faire quelque chose en fonction de ces coordonnées
Fin de si
```

Associer des objets HTML à des expressions 4D

Il est possible d’insérer des **expressions 4D** (et non uniquement des variables) dans les commentaires HTML 4D à l’aide des balises 4DVAR et 4DHTMLVAR.
Ainsi, vous pouvez insérer directement le contenu d’un champ (par exemple <!--4DVAR [nomTable]nomChamp-->) ou d’un élément de tableau (par exemple <!--4DVAR tab{1}-->).

La conversion de l’expression obéit aux mêmes règles que la conversion d’une variable. En outre, l’expression doit satisfaire aux règles de syntaxe de 4D.

Bien qu’une expression puisse contenir directement des appels de fonctions 4D, il est déconseillé d’utiliser ce mode de fonctionnement, pour des raisons de localisation. Par exemple le commentaire <!--4DVAR Date du jour-->, bien que correctement interprété avec un 4D français, ne sera pas traité avec un 4D anglais. Il est en de même pour le traitement des nombre réels (car le séparateur décimal peut varier d’un pays à l’autre). Dans ces deux cas, il est préférable d’affecter une variable par programmation.

En cas d’erreur d’évaluation, le texte inséré sera de la forme “<!--4DVAR mavar--> : ## erreur # code d’erreur”.

Notes:
• Il est possible d'afficher le contenu d'un champ image. En outre (en mode contextuel uniquement), vous pouvez également afficher le contenu d'une variable image.
• En revanche, dans les deux modes, il n'est pas possible d'afficher le contenu d'un élément de tableau image.

Pour plus d'informations, reportez-vous à la section "Services Web, Balises HTML 4D".

Références de fichiers et URLs (mode contextuel)

En mode contextuel, pour assurer la maintenance du contexte de la base et le numéro d'identification du sous-contexte, 4D recalcule automatiquement les références de fichiers et les URLs.

Par exemple, toutes les références IMG et HREF des fichiers locaux sont recalculées. Si vous insérez votre propre code HTML dans un formulaire 4D avec une variable Texte, vous devez employer la syntaxe décrite plus bas.

Les fichiers locaux GIF sont par exemple recalculés en "/4DBin/_/GIF_file_pathname" où GIF_file_pathname est le nom de chemin complet en HTML vers un fichier GIF, compte tenu de la racine du volume où le fichier est situé.

Une petite méthode 4D comme celle qui suit retourne les références recalculées pour le nom de chemin d'accès recu en paramètre :

```
   ` Méthode projet WWW Local GIF URL
   ` WWW Local GIF URL ( Texte )
   ` WWW Local GIF URL ( Chemin d'accès natif ) -> URL vers fichier GIF local
C_TEXTE($0;$1)
$0:="/4DBin/_/"+HTML Pathname ($1)
```

Note : Reportez-vous aux exemples de la commande Mac vers ISO pour plus de détails sur la méthode HTML Pathname.

Ensuite, quand vous insérez le code HTML dans un formulaire 4D au moyen d'une variable Texte, vous pouvez écrire :

```
vtHTML:=Caractere(1)+"<P><IMG SRC="+Caractere(34)+
                                    WWW Local GIF URL("F:.HTM"+Caractere(34)+
                                          " ALIGN=MIDDLE></P>"+Caractere(13)
```

Cette méthode insèrera le document GIF dans le formulaire 4D à l'endroit où la variable 4D vtHTML est placée.

Important : Vous devez écrire ce type de code uniquement si vous insérez du code HTML dans un formulaire 4D. Si vous envoyez simplement une page HTML avec ENVOYER FICHIER HTML ou ENVOYER BLOB HTML, ou encore si vous utilisez une commande telle que AJOUTER ENREGISTREMENT, souvenez-vous que 4D effectue automatiquement la traduction et le recalcul pour vous.

Le recalcul ne change pas les Liens pour les protocoles suivants :
- http:
- ftp:
- mailto:
- news:
- gopher:
- javascript:
- telnet:
- nntp:
- wais:
- prospero:

Référence

ENVOYER BLOB HTML, ENVOYER FICHIER HTML, Services Web, Mode sans contexte.

Le serveur Web 4D propose plusieurs URLs et actions de formulaires HTML spéciaux vous permettant d'effectuer différentes actions dans votre base de données, en mode contextuel ou en mode sans contexte.

Ces URLs sont les suivants :
• 4DMETHOD/, permettant de lier un objet HTML à une méthode projet de votre base en mode contextuel,
• 4DACTION/, permettant de lier un objet HTML à une méthode projet de votre base en mode sans contexte,
• 4DCGI/, permettant d'appeler la Méthode base Sur connexion Web depuis un objet HTML.
• Cette section décrit également la méthode *COMPILER_WEB* ainsi que les déclarations des paramètres de type Texte passés aux méthodes 4D appelées via des URLs.

Note : Par ailleurs, le serveur Web 4D accepte quatre URLs particuliers : /4DSTATS, /4DHTMLSTATS, /4DCACHECLEAR et /4DWEBTEST, vous permettant d'obtenir diverses informations sur le fonctionnement de votre site Web 4D. Ces URLs sont décrits dans la section Services Web, Informations sur le site Web.

URL 4DMETHOD/

Syntaxe : 4DMETHOD/MaMéthode{/Param}
Mode : Contextuel. Appelé depuis le mode sans contexte, provoque le passage en mode contextuel.
Utilisation : URL ou Action de formulaire.

Cet URL vous permet de lier un objet HTML (texte, bouton...) à une méthode projet 4D. Le lien sera du type /4DMETHOD/Nom_Méthode/Param où Nom_Méthode est le nom de la méthode projet 4D à exécuter lorsque l'objet HTML sera activé (par exemple lorsqu'il reçoit un clic souris) et Param un paramètre optionnel de type Texte passé à la méthode dans $1 (reportez-vous ci-dessous au paragraphe “Les paramètres Texte passés aux méthodes via des URLs”). Les éléments liés provoquent l'exécution de la méthode projet 4D à travers leurs URLs. La méthode projet peut elle-même afficher des formulaires 4D, d'autres pages, etc.

Si vous affectez /4DMETHOD/Nom_Méthode en tant qu'action de formulaire à une page HTML statique, la méthode est exécutée lorsque l'utilisateur clique sur un bouton **Submit** dans le formulaire HTML. Reportez-vous à l'exemple de la commande ENVOYER FICHIER HTML. Lorsque vous intégrez des pages HTML dans 4D, vous utilisez généralement des boutons de type Normal ou Submit.

Pour définir l'action d'un formulaire, la syntaxe HTML à employer est de la forme suivante :

```
<FORM ACTION="/4DMETHOD/NomMéthode" METHOD=POST>
```

Des exemples détaillés sont fournis dans la section Services Web, Premiers pas (Partie II).

URL 4DACTION/

Syntaxe : 4DACTION/MaMéthode{/Param}
Mode : Sans contexte. Appelé depuis le mode contextuel, provoque la fermeture du process Web et le passage en mode sans contexte.
Utilisation : URL ou Action de formulaire.

Lorsque 4D reçoit une requête /4DACTION/MAMETH/PARAM, la Méthode base Sur authentification Web (si elle existe) est appelée. Si elle retourne Vrai, la méthode MAMETH est exécutée avec comme paramètre (dans $1) la chaîne /PARAM. La syntaxe de l'URL doit être de la forme suivante :

```
<A HREF="/4DACTION/MAMETH/PARAMS">Faire Quelque Chose</A>
```

En mode contextuel, lorsque 4D reçoit une requête de type /4DACTION, le contexte est tué et la méthode est exécutée hors contexte. C'est précisément l'inverse de /4DMETHOD appelée depuis le mode sans contexte.

Note : Une méthode appelée par 4DACTION ne doit pas faire appel à des éléments d'interface (DIALOGUE, ALERTE...)

Exemple

Vous insérez dans une page HTML statique les instructions suivantes :

```
<IMG SRC="/4DACTION/PICTFROMLIB/1000">
```

Le code de *PICTFROMLIB* est le suivant :

```
C_TEXTE($1) `Ce paramètre doit toujours être déclaré
C_IMAGE($VarImage)
C_BLOB($VarBlob)
C_ENTIER LONG($Numéro)
   `On récupère le numéro d'image dans la chaîne $1
$Numéro:=Num(Sous chaine($1;2;99))
LIRE IMAGE DANS BIBLIOTHEQUE($Numéro;$VarImage)
IMAGE VERS GIF($VarImage;$VarBlob)
ENVOYER BLOB HTML ($VarBlob;"image/gif")
```

4DACTION pour poster des formulaires

Le mode sans contexte permet une facilité supplémentaire lorsque vous souhaitez utiliser des formulaires "postés", c'est-à-dire des pages HTML statiques renvoyant des données au serveur Web. L'action du formulaire doit impérativement débuter par /4DACTION/NomMéthode.

Note : Un formulaire peut être soumis dans deux modes (4D accepte les deux) :
• POST, généralement utilisé pour envoyer des données vers le serveur Web - dans une base de données.
• GET, généralement utilisé pour interroger serveur Web - données en provenance de la base.

Dans ce cas, lorsque le serveur Web reçoit un formulaire posté, il appelle la méthode projet **COMPILER_WEB** (si elle existe), puis la Méthode base Sur authentification Web (si elle existe). Si elle retourne Vrai, la méthode NomMéthode est exécutée. 4D analyse les champs HTML présents dans le formulaire, récupère leurs valeurs et remplit automatiquement des variables 4D avec leur contenu. Il suffit pour cela que les champs du formulaire et les variables 4D portent le même nom.

Note : Vous pouvez également utiliser la commande LIRE VARIABLES FORMULAIRE WEB, permettant de récupérer le non et la valeur de tous les champs inclus dans une page HTML soumise au serveur.

La syntaxe HTML à appliquer dans le formulaire est du type suivant :
• pour la définition de l'action du formulaire :
<FORM ACTION="/4DACTION/NomMéthode" METHOD=POST>
• pour la définition d'un champ du formulaire :
<INPUT TYPE=Type de champ NAME=nom du champ VALUE="Valeur par défaut">

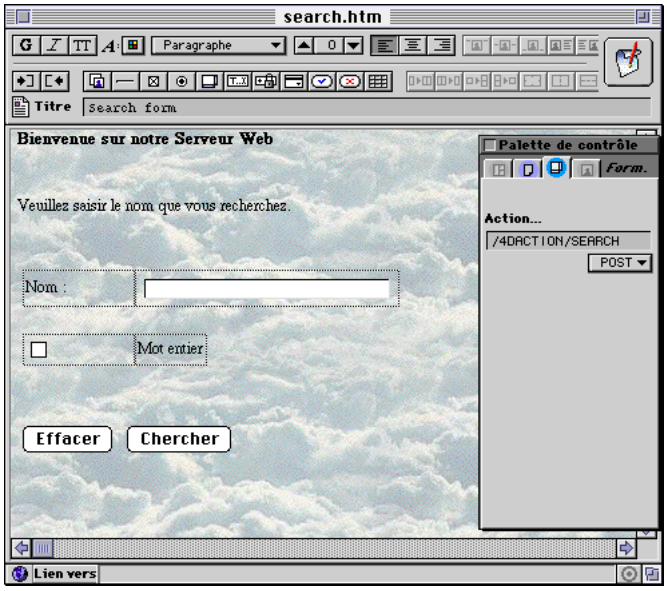
Pour chaque champ du formulaire, 4D affecte la valeur du champ à la variable de même nom.
Pour les options de formulaires (par exemple les cases à cocher), 4D affecte la valeur 1 à la variable associée s'il est coché, sinon 0.
Pour les saisies de type numérique, 4D effectue une conversion Alpha–>Numérique de la valeur du champ.

Note : Si un champ du formulaire est nommé OK (par exemple un bouton Submit), la variable système *OK* de 4D est mise à 1 si la valeur du champ est non vide, sinon 0.

Exemple

Dans une base Web 4D démarrée et utilisée en mode "sans contexte", nous souhaitons que les browsers puissent effectuer des recherches parmi les enregistrements par l'intermédiaire d'une page HTML statique. Cette page s'intitule "search.htm". La base contient d'autres pages statiques, permettant par exemple d'afficher le résultat de la recherche ("results.htm"). Le type POST a été associé à la page, ainsi que l'action /4DACTION/SEARCH.

Voici la page telle qu'elle apparaît dans un éditeur HTML, ici Adobe® PageMill™ :



Voici le code HTML correspondant à cette page :

```
<FORM ACTION="/4DACTION/PROCESSFORM" METHOD=POST>
<INPUT TYPE=TEXT NAME=VNAME VALUE=""><BR>
<!-- Habituellement on met le nom du bouton dans VALUE, pour des raisons d'interprétation il faut mettre un nombre dans VALUE-->
<INPUT TYPE=CHECKBOX NAME=EXACT VALUE="1">Mot entier<BR>
<!-- OK est un cas particulier-->
<INPUT TYPE=SUBMIT NAME=OK VALUE="Chercher">
</FORM>
```

En cours d'utilisation, vous tapez "ABCD" dans la zone de saisie, vous cochez l'option et validez en cliquant sur **Chercher**.
4D appelle alors la méthode projet COMPILER_WEB. Voici son contenu :

```
C_TEXTE(VNAME)
VNAME:=""
C_ENTIER LONG(vEXACT)
vEXACT:=0
OK:=0 `cas particulier
```

Dans l'exemple, *VNAME* contient la chaîne "ABCD", *vEXACT* a pour valeur 1, et *OK* a pour valeur 1 (car on a cliqué sur un bouton dont le nom est OK).

4D appelle la Méthode base Sur authentification Web (si elle existe), puis la méthode projet *PROCESSFORM*, dont voici le contenu :

```
Si (OK=1)
   Si (vEXACT=0) `Si l'option n'a pas été cochée
      vNAME:=VNAME+"@"
   Fin de si
   CHERCHER([Jockeys];[Jockeys]Nom=vNAME)
   vLIST:=Caractere(1)   `Retourne la liste en HTML
   DEBUT SELECTION([Jockeys])
   Tant que (Non(Fin de selection([Jockeys])))
      vLIST:=vLIST+[Jockeys]Nom+" "+[Jockeys]Tél+"<BR>"
      ENREGISTREMENT SUIVANT([Jockeys])
   Fin tant que
   ENVOYER FICHIER HTML("results.htm")   `Envoi de la liste dans le
            `formulaire results.htm, qui contient une référence à la variable vLIST,
            `par exemple <!--4DVAR vLIST-->
   ...
Fin de si
```

URL 4DCGI/

Syntaxe : 4DCGI/<action>
Mode : Tous.
Utilisation : URL.

Lorsque le serveur Web 4D reçoit l'URL /4DCGI/<action>, il appelle la Méthode base Sur connexion Web en passant l'URL "tel quel" dans $1.

L'URL 4DCGI/ ne correspond à aucun fichier. Il a pour unique rôle d'appeler 4D. Le paramètre "<action>" peut contenir n'importe quel type d'information.

Cet URL vous permet donc d'effectuer tout type de traitement. Il vous suffit de tester la valeur de $1 dans la Méthode base Sur connexion Web ou dans une de ses sous-méthodes et d'effectuer dans 4D le traitement adéquat. Par exemple, vous pouvez construire des pages HTML statiques entièrement personnalisées d'ajout, de recherche ou de tri d'enregistrements, ou encore générer des images GIF à la volée. Des exemples d'utilisation de cet URL sont fournis dans les descriptions des commandes IMAGE VERS GIF et ENVOYER REDIRECTION HTTP.

A l'issue du traitement, une "réponse" doit être retournée, à l'aide d'une des commandes d'envoi de données (ENVOYER FICHIER HTML, ENVOYER BLOB HTML, etc.).

ATTENTION : Veillez à effectuer les traitements les plus courts possibles, afin de ne pas bloquer le browser.

Méthode projet COMPILER_WEB

Lorsque le serveur Web 4D reçoit un formulaire posté, il appelle la méthode projet nommée COMPILER_WEB (si elle existe). Cette méthode doit contenir toutes les directives de typage et/ou d'initialisation des variables, en l'occurrence les variables dont les noms sont ceux des champs du formulaire. Elle sera utilisée par 4D Compiler en cas de compilation de la base.
La méthode COMPILER_WEB est commune à tous les formulaires.

Note : Vous pouvez également utiliser la commande LIRE VARIABLES FORMULAIRE WEB, permettant de récupérer le non et la valeur de tous les champs inclus dans une page HTML soumise au serveur.

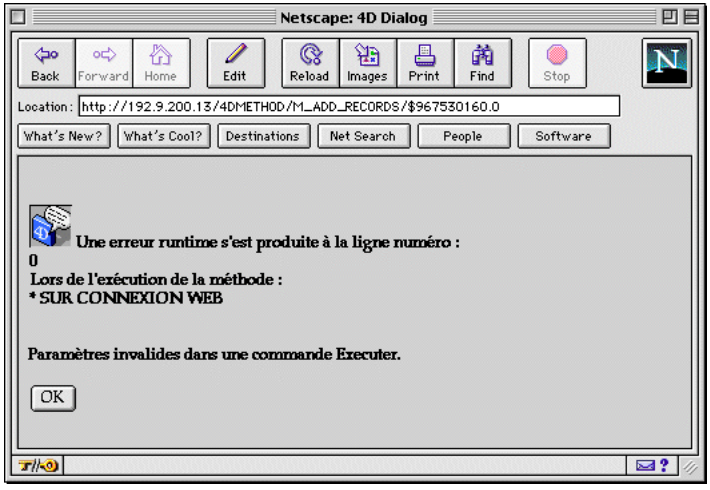
Paramètres texte passés aux méthodes 4D appelées via des URLs

4e Dimension envoie des paramètres Texte aux méthodes 4D appelées par des URLs spéciaux (4DMETHOD/, 4DACTION/ et 4DCGI/), en mode contextuel et en mode sans contexte. Voici quelques remarques sur ces paramètres :

• Même si vous n'utilisez pas ces paramètres, vous devez les déclarer explicitement avec la commande C_TEXTE, sinon des erreurs runtime se produiront lorsque vous accèderez par le Web à une base exécutée en mode compilé.
• Le paramètre $1 retourne des données supplémentaires placées à la fin de l'URL, et peuvent être utilisés pour passer des données de l'environnement HTML vers l'environnement 4D.

Erreurs runtime en mode compilé

Prenons un exemple : vous exécutez une méthode liée à un objet HTML. Vous obtenez l'écran suivant dans votre browser Web :



Cette erreur runtime est provoquée par l'absence de déclaration du paramètre texte $1 dans la méthode 4D appelée lorsque vous avez cliqué sur le lien HTML. Comme le contexte de l'exécution est la page HTML courante, l'erreur fait référence à la “ligne 0” de la méthode qui a envoyé la page au browser Web.

Pour reprendre l'exemple de la section Services Web, Premiers pas (Partie I), le paramètre texte $1 est déclaré explicitement dans les méthodes M_ADD_RECORDS et M_LIST_RECORDS :

```
   `Méthode projet M_ADD_RECORDS
⇒  C_TEXTE($1) ` Ce paramètre DOIT être explicitement déclaré
   Repeter
      AJOUTER ENREGISTREMENT ([Clients])
   Jusque (OK=0)

   `Méthode projet M_LIST_RECORDS
⇒  C_TEXTE($1) ` Ce paramètre DOIT être explicitement déclaré
   TOUT SELECTIONNER ([Clients])
   MODIFIER SELECTION([Clients])
```

Lorsque que ces déclarations sont effectuées, les erreurs runtime en mode compilé ne se produisent pas.

Paramètres à déclarer explicitement dans la méthode appelée

Vous devez déclarer différents paramètres en fonction de l'origine et de la nature de l'appel de la méthode 4D.

• Méthode base Sur authentification Web (si elle existe) et Méthode base Sur connexion Web (appelée, par exemple, par l’URL 4DCGI/<action>)
Vous devez déclarer les six paramètres de la connexion :

```
   ` Méthode base Sur connexion Web
⇒  C_TEXTE($1;$2;$3;$4;$5;$6) ` Ces paramètres DOIVENT être explicitement déclarés
```

• Méthode appelée par l'URL 4DMETHOD/
Vous devez déclarer le paramètre $1 :

```
   ` Méthode appelée par l'URL 4DMETHOD/
⇒  C_TEXTE($1) ` Ce paramètre DOIT être explicitement déclaré
```

• Méthode appelée par l’URL 4DACTION/
Vous devez déclarer le paramètre $1 :

```
   ` Méthode appelée par l'URL 4DACTION/
⇒  C_TEXTE($1) ` Ce paramètre DOIT être explicitement déclaré
```

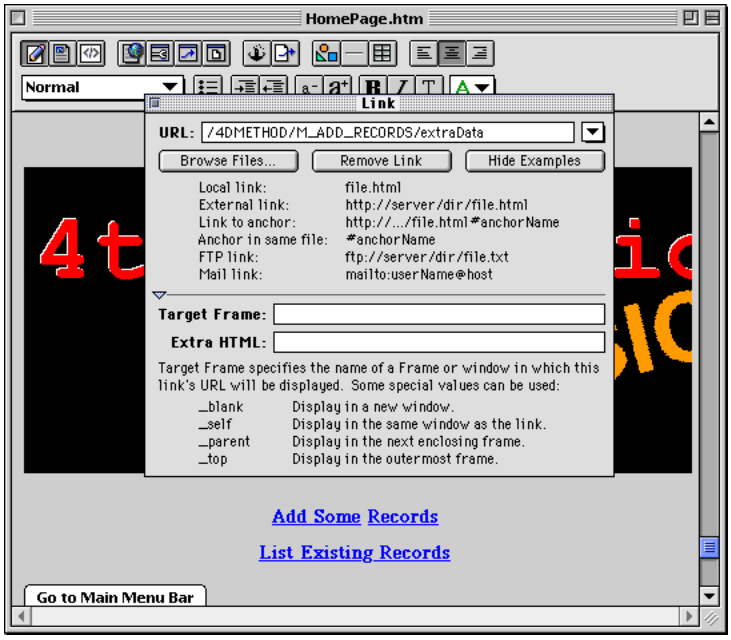
• Méthode appelée par la balise 4DSCRIPT/
La méthode retourne une valeur dans $0. Vous devez déclarer les paramètres $0 et $1 :

```
      ` Méthode appelée par 4DSCRIPT/ sous forme de commentaire HTML
⇒     C_TEXTE($0; $1) ` Ces paramètres DOIVENT être explicitement déclarés
```

Exploiter les données supplémentaires de l'URL

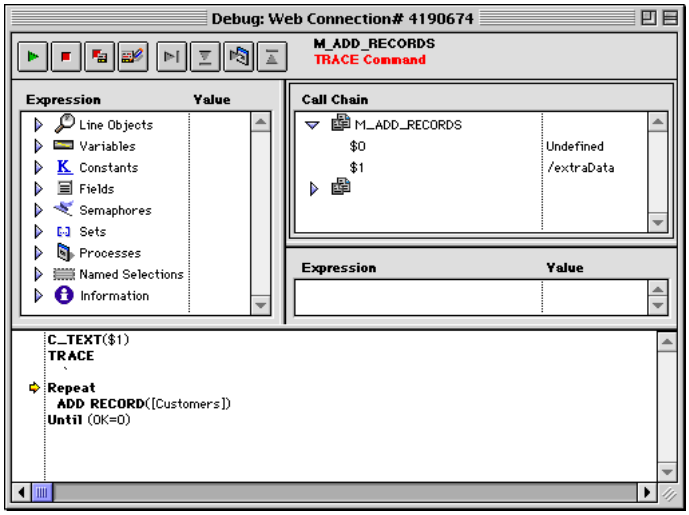
Le paramètre texte $1 passé aux méthodes 4D retourne des données supplémentaires ajoutées à l'URL.

Toujours en suivant l'exemple de la section Services Web, Premiers pas (Partie I), la modification présentée ci-dessous est apportée au lien URL qui référence la méthode M_ADD_RECORDS :



Note : Cette copie d'écran illustre la modification effectuée en version anglaise à l'aide de Claris Home Page sous MacOS.

Les données ajoutées à l'URL sont donc la chaîne “/extraData”. Après que cette modification ait été effectuée, vous pouvez ouvrir la fenêtre du Débogueur, côté 4D, pour vérifier rapidement que le paramètre $1 retourne effectivement la chaîne “/extraData”:



En faisant appel à des conventions et à des algorithmes tels que ceux qui sont décrits dans la section Méthode base Sur connexion Web, vous avez les moyens d'échanger des données supplémentaires entre les environnements HTML et 4D lorsqu'une méthode 4D est appelée par un lien HTML.

Ecrire dynamiquement les données supplémentaires de l'URL

Si vous créez et remplissez vos propres fichiers HTML “à la volée” (en utilisant, par exemple, Creer document et ENVOYER PAQUET), il vous suffit simplement d'écrire les URLs en fonction de vos besoins.
Si vous travaillez avec des fichiers HTML existants, vous pouvez utiliser JavaScript pour écrire dynamiquement la ou les propriété(s) des liens de vos objets.

Référence

Encapsulation HTML et Javascript, LIRE VARIABLES FORMULAIRE WEB, Services Web, Premiers pas (Partie I), Services Web, Premiers pas (Partie II).

Le serveur Web de 4D fournit diverses balises HTML spécifiques à 4D, permettant d'insérer des références à des expressions ou variables 4D, ou différents types de traitements, à l'intérieur des pages HTML statiques envoyées par le serveur Web 4D. Ces pages sont appelées pages semi-dynamiques.
Ces balises sont insérées sous forme de commentaires HTML (<!--LaBalise LeContenu--> en code HTML). La plupart des éditeurs HTML proposent des fonctions facilitant l'insertion de commentaires dans les pages statiques.

Les balises HTML 4D disponibles sont les suivantes :
• 4DVAR, pour insérer des variables et expressions 4D
• 4DHTMLVAR, semblable à 4DVAR mais insérant du code HTML
• 4DSCRIPT, pour exécuter une méthode 4D
• 4DINCLUDE, pour insérer une page HTML dans une autre page
• 4D, 4DELSE et 4DENDIF, pour insérer des conditions dans le code HTML
• 4DLOOP et 4DENDLOOP, pour insérer des boucles dans le code HTML

Traitement des balises HTML 4D

Voici un tableau récapitulatif des cas dans lesquels 4D analyse les balises contenues dans les pages HTML envoyées aux browsers Web :

Conditions d'envoi	**Analyse du contenu des pages envoyées**	
	Mode contextuel	**Mode sans contexte**
• Extensions des pages (cas général) :		
.htm, .html, .shtm, .shtml (HTML pages)	X	X
.xml, .xsl (XML pages)	X	X
.wml (WML pages)	X	X
• Pages appelées par des URLs	X	X, sauf pages suffixées ".htm" ou ".html"
• Commande ENVOYER FICHIER HTML	X	X
• Commande ENVOYER BLOB HTML (si le BLOB est du type "text/html")	X	X
• Inclusion par la balise <!--4DINCLUDE-->	X	X
• Inclusion par la balise {mapage.htm}	X	-

En vue d'une exploitation par 4D, un commentaire HTML doit toujours être de la forme <!--4D...-->. Attention, certains éditeurs HTML ajoutent automatiquement d'autres informations au sein d'un commentaire, ce qui peut entraîner des erreurs d'analyse.
La présence d'autres commentaires HTML (par exemple <!--Début de liste--> est toutefois possible.

Si un commentaire <!--4D... ne se termine pas par -->, un texte de la forme “<!--4D... : --> attendu” sera inséré et l’analyse sera interrompue à ce niveau (la page sera tout de même envoyée pour indiquer l’erreur).

Il est possible d’imbriquer les différents types de commentaires. Voici par exemple une structure HTML parfaitement envisageable :

```
<HTML>
...
<BODY>
<!--4DSCRIPT/PRE_PROCESS-->                    (Appel de méthode)
<!--4DIF (mavar=1)-->                          (Si condition)
      <!--4DINCLUDE banner1.html-->            (Insertion d’une sous page)
<!--4DENDIF-->                                 (Fin de si)
<!--4DIF (mavar=2)-->
      <!--4DINCLUDE banner2.html-->
<!--4DENDIF-->

<!--4DLOOP [TABLE]-->                          (Boucle sur la sélection courante)
<!--4DIF ([TABLE]ValNum>10)-->                 (Si [TABLE]ValNum>10)
      <!--4DINCLUDE subpage.html-->            (Inclusion d'une sous page)
<!--4DELSE-->                                  (Sinon)
      <B>Valeur : <!--4DVAR [TABLE]ValNum--></B><BR>
                                               (Affichage d’un champ)
<!--4DENDIF-->
<!--4DENDLOOP-->                               (Fin de boucle)
</BODY>
</HTML>
```

Balise 4DVAR

Syntaxe : <!--4DVAR NomVar--> ou <!--4DVAR Expression4D-->

La balise <!--4DVAR NomVar--> vous permet d’insérer une référence à une variable ou à une expression 4D à tout endroit d'une page HTML. Par exemple, si vous écrivez :
<P>Bienvenue sur <!--4DVAR vtNomSite-->!</P>
La valeur de la variable 4D vtNomSite sera insérée dans la page HTML lors de son envoi.

Vous pouvez également insérer du code HTML dans une variable texte 4D. Pour cela, il vous suffit de faire débuter la variable par le code ASCII 1 (par exemple, vtHTML:=Caractere(1)+"...code HTML..."). Un autre moyen consiste à utiliser la balise 4DHTMLVAR.

Il est aussi possible d’insérer des *expressions* 4D dans les commentaires HTML 4D à l’aide de la balise 4DVAR. Vous pouvez par exemple insérer directement le contenu d’un champ (<!--4DVAR [nomTable]nomChamp-->) ou un élément de tableau (<!--4DVAR tab{1}-->).
La conversion de l’expression obéit aux mêmes règles que la conversion d’une variable. En outre, l’expression doit satisfaire aux règles de syntaxe de 4D.

Bien qu'une expression puisse contenir directement des appels de fonctions 4D, il est déconseillé d'utiliser ce mode de fonctionnement, pour des raisons de localisation. Par exemple le commentaire <!--4DVAR Date du jour-->, bien que correctement interprété avec un 4D français, ne sera pas traité avec un 4D anglais. Il est en de même pour le traitement des nombres réels (car le séparateur décimal peut varier d'un pays à l'autre). Dans ces deux cas, il est préférable d'affecter une variable par programmation.

En cas d'erreur d'évaluation, le texte inséré sera de la forme "<!--4DVAR mavar--> : ## erreur # code d'erreur".

Notes :
• Vous devez utiliser des variables process.
• Il est possible d'afficher le contenu d'un champ image. En outre (en mode contextuel uniquement), vous pouvez également afficher le contenu d'une variable image. En revanche, dans les deux modes, il n'est pas possible d'afficher le contenu d'un élément de tableau image.
• Comme le HTML est orienté texte, vous travaillerez généralement avec des variables Texte. Toutefois, vous pouvez également utiliser des variables BLOB (ce qui vous permet de vous affranchir de la limite des 32 000 caractères inhérente aux variables de type texte). Il vous suffit pour cela de générer des BLOBs en mode Texte sans longueur.
• Des exemples d'utilisation de la balise 4DVAR sont fournis dans la section Encapsulation HTML et Javascript.

Note de compatibilité : Dans les versions 6.0.x de 4D, l'insertion de variables 4D dans les pages statiques s'effectuait à l'aide d'une notation utilisant des crochets ([NomVar]). Si vous travaillez avec une base créée avec une version précédente de 4D et souhaitez utiliser la notation standard <!--4DVAR NomVar-->, assurez-vous que l'option "Utiliser les commentaires 4DVAR au lieu des crochets" dans la boîte de dialogue des Propriétés de la base est bien cochée (reportez-vous à la section Services Web, Paramétrages du serveur Web).

Balise 4DHTMLVAR

Syntaxe : <!--4DHTMLVAR NomVar--> ou <!--4DHTMLVAR Expression4D-->

Cette balise permet d'évaluer une variable ou une expression 4D et de l'insérer dans une page en tant qu'expression HTML. En fait, cette balise produit exactement les mêmes effets que la balise <!--4DVAR mavar-->, lorsque *mavar* débute par le caractère de code ASCII 1.

Par exemple, voici les résultats de l'insertion de la variable Texte 4D *mavar* à l'aide des balises disponibles :

Valeur de *mavar*	**Balises**	**Insertion dans la page Web**
mavar:="<B>"	<!--4DVAR mavar-->	<B>
mavar:=Caractere(1)+"<B>"	<!--4DVAR mavar-->	<B>
mavar:="<B>"	<!--4DHTMLVAR mavar-->	<B>

En cas d'erreur d'évaluation, le texte inséré sera de la forme "<!--4DHTMLVAR mavar--> : ## erreur # code d'erreur".

Note : La variable doit être exprimée à l'aide de la table de caractères ISO Latin-1 (pour plus d'informations, reportez-vous à la description de la commande Mac vers ISO).

Balise 4DSCRIPT/

Syntaxe : <!--4DSCRIPT/NomMéthode/Param-->

La balise 4DSCRIPT/ vous permet d'exécuter des méthodes 4D au moment de l'envoi de pages HTML statiques. La présence dans une page statique du commentaire HTML <!--4DSCRIPT/NomMéthode/Param--> provoque l'exécution de la méthode NomMéthode avec le paramètre *Param* sous forme de chaîne dans $1. Au chargement de la page, 4D appelle la Méthode base Sur authentification Web (si elle existe). Si elle retourne Vrai, 4D exécute la méthode.
La méthode retourne alors du texte dans $0. Si la chaîne débute par le caractère de code ASCII 1, elle est considérée comme du HTML (même principe que pour les variables).

Note : Pour que ce mécanisme fonctionne, il est nécessaire que l'option "Utiliser les commentaires 4DVAR au lieu des crochets" soit cochée dans les propriétés de la base (cf. section Services Web, Paramétrages du serveur Web).

L'analyse du contenu de la page s'effectue au moment de l'appel à ENVOYER FICHIER HTML (.htm, .html, .shtm, .shtml) ou ENVOYER BLOB HTML (blob de type text/html).
Rappelons qu'en mode sans contexte, l'analyse s'effectue également lorsqu'un URL pointe sur un fichier suffixé ".shtm" ou ".shtml" (par exemple http://www.server.com/dir/page.shtm).

Note : En mode contextuel, la méthode est exécutée dans le contexte.

Par exemple, vous insérez dans une page statique le commentaire "Nous sommes le <!--4DSCRIPT/MAMETH/MONPARAM-->". Au chargement de la page, 4D appelle la Méthode base Sur authentification Web (si elle existe) puis la méthode MAMETH en lui passant comme paramètre (dans $1) la chaîne "/MONPARAM".
La méthode retourne du texte dans $0 (par exemple "28/10/99"), l'expression "Nous sommes le <!--4DSCRIPT/MAMETH/MONPARAM-->" devient alors "Nous sommes le 28/10/99".

Le code de *MAMETH* est :

```
C_TEXTE($0;$1) `Ces paramètres doivent TOUJOURS être déclarés
$0:=Chaine(Date du jour)
```

Note : Une méthode appelée par 4DSCRIPT ne doit pas faire appel à des éléments d'interface (DIALOGUE, ALERTE...)

Comme 4D exécute les méthodes dans leur ordre d'apparition, il est tout à fait possible de faire appel à une méthode qui fixe la valeur de plusieurs variables elles-mêmes référencées plus loin dans le document, quel que soit le mode utilisé.

Note : Vous pouvez insérer autant de commentaires <!--4DSCRIPT...--> que vous voulez dans une page statique.

Note de compatibilité : Dans les versions précédentes de 4D, la même balise, 4DACTION, pouvait être utilisée soit sous forme d'URL (par exemple http://monserver/4DACTION/meth), soit sous forme de commentaire HTML dans une page statique (<!--4DACTION/meth-->). Cette diversité d'utilisations pouvant entraîner des confusions, la balise de commentaire 4DSCRIPT permet désormais de dissocier les deux usages. La balise 4DSCRIPT s'utilise exclusivement sous forme de commentaire HTML (<!--4DSCRIPT/meth-->) et produit exactement les mêmes effets que <!--4DACTION/meth-->. 4DACTION est désormais réservée à une utilisation en tant qu'URL.

Balise 4DINCLUDE

Syntaxe : <!--4DINCLUDE chemin-->

Ce nouveau commentaire permet d'inclure, dans une page HTML, le corps d'une autre page HTML (désignée par le paramètre chemin). Le corps d'une page HTML désigne ce qui est compris entre les balises <BODY> et </BODY> (les balises elles-mêmes ne sont pas incluses).
Le commentaire <!--4DINCLUDE --> s'avère particulièrement utile en combinaison avec les tests (<!--4DIF-->) ou les boucles (<!--4DLOOP-->). Il est également pratique pour inclure des bannières en fonction d'un critère, ou de manière aléatoire.
Au moment de l'inclusion, quels que soient le mode et l'extension du nom du fichier, 4D analyse la page appelée puis insère son contenu — éventuellement modifié — dans la page d'où provient l'appel 4DINCLUDE.

Le placement dans le cache Web d'une page incluse à l'aide du commentaire <!--4DINCLUDE --> répond aux mêmes règles que celles des pages demandées via un URL ou envoyées par la commande ENVOYER FICHIER HTML.

Passez dans chemin le chemin d'accès au document à inclure. Le chemin d'accès est relatif au document en cours d'analyse. Utilisez la barre oblique (/) comme séparateur de dossiers et les deux-points (..) pour remonter d'un niveau hiérarchique (syntaxe HTML).

Le nombre de <!--4DINCLUDE chemin--> au sein d'une page n'est pas limité. Toutefois, les appels à <!--4DINCLUDE chemin--> ne peuvent s'effectuer que sur 1 niveau. Cela signifie que par exemple vous ne pouvez pas insérer le commentaire <!--4DINCLUDE mondoc3.html--> dans le corps de la page mondoc2.html, elle-même appelée par <!--4DINCLUDE mondoc2--> inséré dans mondoc1.html.
En outre, 4D contrôle que les inclusions ne sont pas récursives.

En cas d'erreur, le texte inséré est de la forme "<!--4DINCLUDE chemin--> : Le document ne peut pas être ouvert".

Note : En mode contextuel, si une page est insérée dans un formulaire via un marqueur {mapage.html} inséré dans une zone de texte statique, les éventuels commentaires 4DINCLUDE qu'elle contient seront ignorés.

Exemples

```
<!--4DINCLUDE souspage.html-->
<!--4DINCLUDE dossier/souspage.html-->
<!--4DINCLUDE ../dossier/souspage.html-->
```

Balises 4DIF, 4DELSE et 4DENDIF

Syntaxe : <!--4DIF expression--> <!--4DELSE--> <!--4DENDIF-->

Utilisé en conjonction avec les commentaires <!--4DELSE--> (optionnel) et <!--4DENDIF-->, le commentaire <!--4DIF expression--> offre la possibilité d'exécuter du code HTML de manière conditionnelle.
Le paramètre expression peut contenir toute expression 4D valide retournant une valeur booléenne. Elle doit figurer entre parenthèses et respecter les règles de syntaxe de 4D.

Les blocs <!--4DIF expression--> ... <!--4DENDIF--> peuvent être imbriqués sur plusieurs niveaux. Comme dans 4D, chaque <!--4DIF expression--> doit avoir un <!--4DENDIF--> correspondant.
En cas d'erreur d'évaluation, le texte "<!--4DIF expression--> : Une expression booléenne était attendue" est inséré à la place du contenu situé entre <!--4DIF --> et <!--4DENDIF-->. De même, s'il n'y a pas autant de <!--4DENDIF--> que de <!--4DIF -->, le texte "<!--4DIF expression--> : 4DENDIF attendu" est inséré à la place du contenu situé entre <!--4DIF --> et <!--4DENDIF-->.

Exemple

Cet exemple de code inséré dans une page HTML statique affiche un libellé différent en fonction du résultat de l'expression vnom#"" :

```
<BODY>
...
<!--4DIF (vnom#"")-->
Noms commençant par <!--4DVAR vnom-->.
<!--4DELSE-->
Aucun nom n'a été trouvé.
<!--4DENDIF-->
...
</BODY>
```

Balises 4DLOOP et 4ENDLOOP

Syntaxe : <!--4DLOOP condition--> <!--4DENDLOOP-->

Ce commentaire permet de répéter une portion de code HTML tant que la condition est remplie. La portion est délimitée par <!--4DLOOP--> et <!--4DENDLOOP-->.
Les blocs <!--4DLOOP condition--> ... <!--4DENDLOOP--> peuvent être imbriqués. Comme dans 4D, chaque <!--4DLOOP condition--> doit avoir un <!--4DENDLOOP--> correspondant.

Il existe trois types de conditions :

• **<!--4DLOOP [table]-->**
Cette syntaxe effectue une boucle pour chaque enregistrement de la sélection courante de table dans le process en cours. La portion de code HTML située entre les deux commentaires est répétée pour chaque enregistrement de la sélection courante.

Note : Lors de l'utilisation de 4DLOOP avec une table, les enregistrements sont chargés en mode Lecture seule.

L'exemple de code HTML suivant :

```
<!--4DLOOP [Personnes]-->
<!--4DVAR [Personnes]Nom--> <!--4DVAR [Personnes]Prenom--><BR>
<!--4DENDLOOP-->
```

... pourrait se traduire en langage 4D par :

```
DEBUT SELECTION([Personnes])
Tant que(Non(Fin de selection([Personnes])))
   ...
   ENREGISTREMENT SUIVANT([Personnes])
Fin tant que
```

• **<!--4DLOOP tableau-->**
Cette syntaxe effectue une boucle pour chaque élément du tableau. L'indice courant du tableau est incrémenté à chaque répétion de la portion de code HTML.
Il n'est pas possible d'utiliser cette syntaxe avec des tableaux à deux dimensions. Dans ce cas, la solution consiste à combiner une méthode des boucles imbriquées.

L'exemple de code HTML suivant :

```
<!--4DLOOP tab_noms-->
<!--4DVAR tab_noms{tab_noms}--><BR>
<!--4DENDLOOP-->
```

... peut se traduire en langage 4D par :

```
Boucle($indice;1;Taille tableau(tab_noms))
   tab_noms:=$indice
   ...
Fin de boucle
```

• **<!--4DLOOP méthode-->**
Cette syntaxe effectue une boucle tant que la méthode retourne Vrai. La méthode admet un paramètre de type Entier long. Elle est appelée une première fois avec la valeur 0 pour permettre une éventuelle phase d'initialisation, puis elle est appelée successivement avec les valeurs 1, 2, 3... tant qu'elle renvoie Vrai.
Pour des raisons de sécurité, la Méthode base Sur authentification Web peut être appelée, une seule fois, avant la phase d'initialisation (exécution de la méthode avec 0 comme paramètre). Si l'authentification est confirmée, la phase d'initialisation a lieu.

En vue de la compilation de la base, il est impératif de déclarer C_BOOLEEN($0) et C_ENTIER LONG($1) au sein de la méthode.

L'exemple de code HTML suivant :

```
<!--4DLOOP ma_methode-->
<!--4DVAR var--> <BR>
<!--4DENDLOOP-->
```

... peut se traduire en code 4D par :

```
Si(AuthentificationWebOK)
   Si(ma_methode(0))
      $compteur:=1
      Tant que(ma_methode($compteur))
         ...
         $compteur:=$compteur+1
      Fin tant que
   Fin de si
Fin de si
```

La méthode *ma_methode* pourrait avoir la forme suivante :

```
C_ENTIER LONG($1)
C_BOOLEEN($0)
Si ($1=0)
      `Initialisation
   $0:=Vrai
Sinon
   Si($1<50)
      ...
      var:= ...
      $0:=Vrai
   Sinon
      $0:=Faux   `Arrêt de la boucle
   Fin de si
Fin de si
```

En cas d'erreur d'évaluation, le texte "<!--4DLOOP expression--> : descriptif" est inséré à la place du contenu situé entre <!--4DLOOP --> et <!--4DENDLOOP-->.

Le descriptif de l'erreur peut être l'un des suivants :
- Une expression de ce type n'était pas attendue (erreur générique).
- Nom de table invalide (erreur sur le nom de la table).
- Un tableau était attendu (la variable n'est pas un tableau ou est un tableau à deux dimensions).
- La méthode n'existe pas.
- Erreur de syntaxe (lors de l'exécution de la méthode).
- Erreur de privilège (pas de droits suffisants pour accéder à la table ou à la méthode).
- 4DENDLOOP attendu (le nombre de <!--4DENDLOOP--> n'est pas égal à celui de <!--4DLOOP -->).

Référence

Encapsulation HTML et Javascript, Services Web, Mode sans contexte, Services Web, URLs et actions de formulaires.

Vous pouvez obtenir diverses informations sur le fonctionnement de votre site Web 4D :

• Vous pouvez contrôler le site par l'intermédiaire d'URL particuliers (/4DSTATS, /4DHTMLSTATS, /4DCACHECLEAR et /4DWEBTEST).
• Vous pouvez générer un historique des requêtes.
• Vous pouvez visualiser la charge du serveur Web dans la page "Evaluation" de l'Explorateur d'exécution de 4D.

URLs de gestion du serveur Web

Le serveur Web 4D accepte quatre URLs particuliers : /4DSTATS, /4DHTMLSTATS, /4DCACHECLEAR et /4DWEBTEST.
• /4DSTATS, /4DHTMLSTATS et /4DCACHECLEAR sont accessibles uniquement au Super_Utilisateur et à l'Administrateur de la base. Si la base ne comporte pas de système de mots de passe 4D, ces URLs sont accessibles à tout utilisateur.
• /4DWEBTEST est toujours accessible.

/4DSTATS

L'URL /4DSTATS renvoie sous forme texte pur :
• le nombre de "hits" (connexions bas niveau),
• le nombre de contextes créés,
• le nombre de contextes n'ayant pas pu être créés,
• le nombre d'erreurs de mots de passe,
• le nombre de pages stockées dans le cache,
• le pourcentage du cache utilisé,
• la liste des pages et des fichiers GIF ou JPEG stockés dans le cache des pages statiques (*).

(*) Pour plus d'informations sur le cache des pages statiques et des images, reportez-vous à la section Services Web, Paramétrages du serveur Web.

Ces informations peuvent vous permettre de contrôler le fonctionnement de votre serveur et éventuellement d'adapter les paramètres correspondants.

Note : La commande STATISTIQUES DU CACHE WEB vous permet également d'obtenir des informations sur l'utilisation du cache pour les pages statiques.

/4DHTMLSTATS

L'URL /4DHTMLSTATS renvoie, sous forme de texte pur également, les mêmes informations que l'URL /4DSTATS, à la différence près que, dans la dernière rubrique (contenu du cache), seule la liste des pages HTML — donc sans les fichiers .GIF et .JPEG — présentes dans le cache est fournie.

/4DCACHECLEAR

L'URL /4DCACHECLEAR provoque l'effacement immédiat du cache des pages statiques et des images. Il permet donc de "forcer" la mise à jour de pages ayant été modifiées.

/4DWEBTEST

L'URL /4DWEBTEST permet de contrôler le statut du serveur Web. Lorsque cet URL est appelé, 4D retourne un fichier texte contenant les champs HTTP suivants :
• **Date** : date du jour au format RFC 822
Exemple : "Date: Wed, 26 Jan 2000 13:12:50 GMT"
• **Server** : 4D WebStar_D/numéro de version
Exemple : "4D WebStar_D/6.7"
• **User-Agent** : nom et version @ adresse IP du client
Exemple : "Mozilla/4.08 (Macintosh; I; PPC, Nav) @ 192.193.00.00"

Historique des requêtes

4D vous permet de générer un historique des requêtes. L'historique se présente sous la forme d'un fichier texte nommé "logweb.txt" automatiquement placé à côté du fichier de structure de la base. Ce fichier est au format CLF (Common LogFile Format) ou format NCSA, reconnu par la plupart des outils d'analyse des sites Web.

Chaque ligne du fichier représente une requête sous la forme :
hôte rfc931 utilisateur [JJ/MMM/AAAA:HH:MM:SS] "requête" statut longueur

Chaque champ est séparé par un espace, chaque ligne se termine par la séquence CR/LF (caractère 13, caractère 10).

• hôte : adresse IP du client (ex. 192.100.100.10)
• rfc931 : information non gérée par 4D, c'est toujours - (signe moins)
• utilisateur : nom de l'utilisateur tel qu'il s'est authentifié, sinon - (signe moins). Si le nom de l'utilisateur contient des espaces, ils sont remplacés par des _ (tiret bas).
• JJ : jour, MMM : mois abrégé en 3 lettres et toujours en anglais (Jan, Feb, ...), AAAA : année, HH : heure, MM : minutes, SS : secondes
La date et l'heure sont locales au serveur
• requête : requête envoyée par le client (ex. GET /index.htm HTTP/1.0)
• statut : réponse donnée par le serveur.
• longueur : taille des données renvoyées (hors en-tête HTTP) ou 0.

Note : Pour des raisons de performances, les opérations sont conservées dans une mémoire tampon par paquets de 1Ko avant d'être écrites sur disque. Les opérations sont également écrites sur disque en cas d'absence de requête pendant 5 secondes.

Les valeurs possibles de statut sont :
200 : OK
204 : Pas de contenu
302 : Redirection
400 : Mauvaise requête
401 : Authentification requise
404 : Non trouvé
500 : Erreur interne

Exemples de lignes générées par l'historique des requêtes :

192.100.100.10 - - [25/Jan/1998:12:54:06] "GET /index.htm" 200 6524
Le client Web à l'adresse 192.100.100.10 ne s'est pas authentifié. Il a demandé la page index.htm, qui a été servie (elle contient 6524 octets).

192.100.101.25 - - [25/Jan/1998:12:54:09] "GET /123456.htm" 404 125
Le client Web à l'adresse 192.100.101.25 ne s'est pas authentifié. Il a demandé la page 123456.htm, qui n'a pas été trouvée (4D a envoyé un message de 125 octets).

192.100.101.31 - - [25/Jan/1998:12:54:10] "GET /secret.htm" 401 0
Le client Web à l'adresse 192.100.101.31 ne s'est pas authentifié. Il a demandé la page secret.htm, le serveur a requis l'authentification.

192.100.101.31 - ZZZZ [25/Jan/1998:12:54:11] "GET /secret.htm" 401 0
Le client Web à l'adresse 192.100.101.31 s'est authentifié comme ZZZZ. Il a demandé la page secret.htm, le nom d'utilisateur est inconnu.

192.100.101.31 - ACI [25/Jan/1998:12:54:12] "GET /secret.htm" 200 2543
Le client Web à l'adresse 192.100.101.31 s'est authentifié comme ACI. Il a demandé la page secret.htm, qui a été servie (elle contient 2543 octets).

ATTENTION : L'historique peut être importé dans un tableur ou dans 4D. Toutefois, il faut impérativement arrêter le serveur Web avant de procéder à l'import.

Par défaut, le fichier d'historique des requêtes n'est pas généré. Pour demander la génération de l'historique des requêtes Web :

1. Dans la page "Serveur Web II" des Propriétés de la base, cochez l'option **Enregistrer requêtes**.

2. Cliquez sur **OK**.

Informations fournies par l'Explorateur d'exécution de 4D

La page Evaluation (rubrique “Informations”) de l’Explorateur d’exécution affiche différentes informations concernant le fonctionnement du serveur Web 4D :

• **Occupation du cache Web** : indique le nombre de pages présentes dans le cache Web ainsi que son pourcentage d’utilisation. Cette information n’est disponible que si le serveur Web est actif et si la taille du cache différente de 0.

• **Temps d’activité du serveur Web** : indique la durée de fonctionnement (au format heures:minutes:secondes) du serveur Web. Cette information n’est disponible que si le serveur Web est actif.

• **Nombre de requêtes http** : indique le nombre total de requêtes HTTP reçues depuis le démarrage du serveur Web, ainsi que le nombre instantané de requêtes par secondes (mesure prise entre deux mises à jour de l’Explorateur d’exécution). Cette information n’est disponible que si le serveur Web est actif.

Référence

Services Web, Paramétrages du Serveur Web, STATISTIQUES DU CACHE WEB.

Le serveur Web 4D permet de tirer parti des CGI (*Common Gateway Interface*). Les CGI sont aux pages Web ce que les plug-ins sont aux méthodes 4D. Appelé par le serveur Web, le CGI exécute une tâche et retourne une réponse — une page Web complète ou du code HTML venant s'insérer dans une page envoyée par le serveur. Des CGI sont fréquemment utilisés pour afficher les compteurs d'accès, gérer les livres d'or (*guestbook*), recevoir le résultat d'un formulaire (*form to mail*), etc.

Note : Le terme CGI désigne, à l'origine, la norme définissant l'interfaçage des applications externes avec les serveurs HTTP. Par extension, "CGI" est aujourd'hui utilisé pour désigner ces applications externes elles-mêmes.

4D peut utiliser des CGI de trois manières :
• le serveur Web 4D peut appeler des CGI
• le serveur Web 4D peut être communiquer directement avec les serveurs Webstar® (MacOS uniquement)
• le serveur Web 4D peut être interrogé par tout serveur HTTP via des extensions de type CGI et ISAPI (Windows uniquement).

Appeler des CGI depuis le serveur Web 4D

Un CGI se présente sous la forme d'un programme exécutable, d'un script PERL ou d'une DLL s'interfaçant avec un serveur Web. Une multitude de CGI sont aujourd'hui disponibles. La plupart d'entre eux sont dans le domaine public.

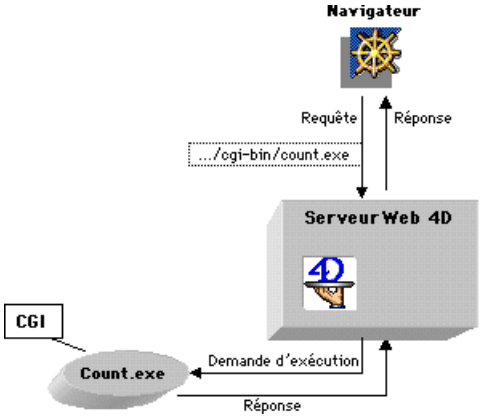
Installer un CGI sur le serveur Web 4D

L'appel d'un CGI s'effectue par l'intermédiaire d'un URL, d'une action ou d'une balise HTML insérée dans une page, en fonction de la tâche effectuée par le CGI. Dans tous les cas, la chaîne HTML doit contenir /cgi-bin/ suivi du nom du CGI et éventuellement d'un chemin d'accès (utilisant la syntaxe HTML) ainsi que d'une chaîne d'interrogation.

Par exemple, l'URL "http://195.1.2.3/cgi-bin/search.exe" provoquera l'exécution du CGI search.exe. De même, si vous placez la balise <IMG SRC="/special/cgi-bin/counter.exe"> au sein d'une page HTML, le CGI counter.exe sera exécuté lors de l'envoi de la page.

Pour pouvoir être appelés, les CGI doivent obligatoirement être placés à la racine d'un dossier nommé **cgi-bin**. Ce dossier doit lui-même être situé à la racine du serveur Web ou dans un sous-dossier. Il peut y avoir plusieurs dossiers **cgi-bin** par serveur. Ce dossier peut contenir d'autres fichiers que des exécutables, mais seuls ces derniers peuvent être appelés depuis un client Web.

Exemple d'installation avec un CGI appelé “Count.exe”:



Voici des exemples d’emplacements et les URL pouvant être appelés :

Emplacement des éléments (racine du serveur Web)	**URLs correspondants**
Dossier [mabase]	
mabase.4db (structure)	(http://195.1.2.3/)
Dossier [cgi-bin]	
compteur.exe	(http://195.1.2.3/cgi-bin/compteur.exe)
Dossier [Divers]	
Dossier [cgi-bin]	
script.pl	(http://195.1.2.3/Divers/cgi-bin/script.pl)

Types de CGI acceptés

Les types de CGI utilisables sont différents sous Windows et sous MacOS.

• **Sous Windows**

Sous Windows, les CGI peuvent être du type suivant :
- des **exécutables** (.EXE) utilisant la “console” et les variables d’environnement. Le code source est généralement multi-plate-forme (Windows et Unix). Leurs noms sont de la forme nnnn.exe ou nph-nnnn.exe. Pour plus d’informations sur ce type de CGI, veuillez consulter le site http://hoohoo.ncsa.uiuc.edu/cgi/.
- des **DLL ISAPI**, c’est-à-dire des extensions pour IIS (*Internet Information Server*). Leurs noms sont de la forme nnnn.dll ou nph-nnnn.dll. Pour des raisons de performances, les DLL appelées de la sorte ne sont déchargées qu’à l’arrêt du serveur Web.
Pour plus d’informations sur ce type de CGI, veuillez consulter le site http://www.microsoft.com/iis/.
- des **scripts PERL** utilisant la “console” et les variables d’environnement. Ces CGI nécessitent la présence d’un interpréteur permettant de les exécuter. Ils présentent toutefois l’avantage d’être entièrement multi-plate-forme (Windows, Unix et MacOS). Leurs noms sont de la forme nnnn.pl, nph-nnnn.pl, nnnn.cgi ou encore nph-nnnn.cgi.
Pour plus d’informations sur ce type de CGI, veuillez consulter le site http://www.perl.com/.

• **Sous MacOS**
Sous MacOS, les CGI peuvent être du type suivant :
- des **applications**, extensions de WebStar et MacHTTP. Leurs noms sont de la forme nnnn.cgi et nnnn.acgi (ce sont obligatoirement des applications).
Pour plus d'informations sur ce type de CGI, veuillez consulter le site http://dev.starnine.com/.
- des **scripts PERL**. Ces CGI nécessitent la présence de MacPERL. Leurs noms sont de la forme nnnn.pl, nph-nnnn.pl, nnnn.cgi ou encore nph-nnnn.cgi (ce sont obligatoirement des fichiers texte).
Pour plus d'informations sur ce type de CGI, veuillez consulter le site http://www.macperl.com/.

Interaction entre le serveur Web 4D et les CGI

L'appel d'un CGI ne modifie jamais l'environnement de 4D (sélections, variables...).
4D n'impose aucune limite de taille de la réponse. Cependant, la durée maximum de traitement allouée au CGI est fixée à 30 secondes. Au-delà de ce délai, le serveur Web retournera une erreur.
Un CGI est toujours exécuté hors contexte, quel que soit le mode depuis lequel il a été appelé.
A noter toutefois qu'en mode contextuel, il est conseillé de ne pas utiliser de CGI qui renvoie du code HTML, car il y a dans ce cas risque de désynchronisation du contexte.

Erreurs renvoyées par 4D lors d'un appel CGI (Windows et MacOS)

Lorsque l'appel d'un CGI génère une erreur, 4D retourne une des réponses suivantes, sous forme de page HTML standard :
• *Non trouvé* : le CGI n'a pu être localisé par 4D, ou bien l'interpréteur PERL n'est pas installé
• *Interdit* : le client Web demande autre chose qu'un exécutable dans un dossier [cgi-bin]
• *Timeout* : le CGI n'a pu traiter la requête en 30 secondes
• *Mauvaise réponse* : la réponse du CGI n'a pu être traitée par 4D ou la DLL ISAPI a causé une exception
• *Erreur interne* : mémoire saturée, etc.

Protection de 4D vis-à-vis d'une DLL ISAPI (Windows uniquement)

Bien qu'appelée directement, si une DLL ISAPI génère une exception (non définie par l'utilisateur), celle-ci n'entraîne pas l'arrêt de 4D. 4D renvoie la réponse "Mauvaise réponse" (la DLL a été incapable de répondre).

Informations à destination des développeurs de CGI

Ce paragraphe est principalement destiné aux programmeurs souhaitant développer des CGI pour leur bases 4D.

• Variables d'environnement

4D définit les variables d'environnement en conformité avec les spécifications CGI/1.1, avec les précisions suivantes :
GATEWAY_INTERFACE : toujours "CGI/1.1"
SERVER_SOFTWARE : toujours de la forme "4D_WebStar_D/version"
SERVER_PROTOCOL : toujours "HTTP/1.0"
SERVER_PORT_SECURE : contient "1" si la connexion HTTP est sécurisée, sinon "0".
PATH_TRANSLATED : contient le chemin d'accès complet de la racine HTML du serveur, auquel est ajouté la portion de chemin qui suit le nom du CGI. Par sécurité, la portion de chemin ne peut contenir les séquences // ou ..
Exemple : Racine du serveur C:/web
Pour un appel CGI du type /cgi-bin/cgi.exe/path, PATH_TRANSLATED vaut "C:/web/path".
Pour un appel CGI du type /cgi-bin/cgi.exe/../path, 4D renvoie l'erreur *Interdit*.
REMOTE_IDENT : nom d'utilisateur, sinon non définie
HTTP_AUTHORIZATION, HTTP_CONTENT_LENGTH et HTTP_CONTENT_TYPE : non définies
ALL_HTTP et URL sont définies dans le cas d'appels de DLL ISAPI.
CERT_xxx et HTTPS_xxx sont définies si la connexion est sécurisée (pour les DLL uniquement).

En plus des variables d'environnement standard, 4D ajoute des variables texte du type FORMVAR_nomvariable :
- si la requête est envoyée avec la méthode "POST", ces variables correspondent aux zones de saisie du formulaire (par exemple FORMVAR_NOM, FORMVAR_PRENOM...) à l'exception des champs binaires (INPUT TYPE="FILE"). Ce système peut être utilisé avec les formulaires encodés "www/url-encoded" et "multipart/form-data".
- si la requête est envoyée avec la méthode "GET", ces variables correspondent aux valeurs passées par la chaîne d'interrogation (par exemple, dans le cas de l'URL .../cgi.exe?nom=martin&code=75, FORMVAR_NOM vaudra "martin" et FORMVAR_CODE vaudra "75").

Ce fonctionnement présente l'avantage de faciliter le traitement des formulaires (il n'est pas nécessaire d'analyser les chaînes a=1&b=2&...), mais rend le CGI spécifique à 4D.

• Traitement des réponses retournées par les CGI

Si le nom du CGI (exécutable Windows ou script PERL) débute par nph- (*No Parsing Header*), 4D retourne la réponse "telle quelle" au client Web. Dans ce cas, il revient au CGI de respecter la norme HTTP. En ce qui concerne les DLL ISAPI, 4D n'analyse jamais la réponse, que le préfixe nph- soit présent ou non.

Si ce n'est pas le cas, 4D se charge de renvoyer l'en-tête HTTP :
- si "Content-Type" n'est pas spécifié par le CGI, 4D renvoie systématiquement "Content-Type: text/html",
- si "Location" est spécifié, 4D ignore les autres éléments de la réponse et effectue une redirection HTTP,

- si “Status” n’est pas spécifié, 4D renvoie “HTTP/1.0 200 OK”.
4D accepte tout type de changement de ligne (Windows-CRLF, MacOS-CR, Unix-LF) dans l’en-tête de la réponse HTTP et se charge de la reformater.

Dans le cas des DLL ISAPI, 4D accepte les traitements asynchrones (HttpExtensionProc retourne HSE_STATUS_PENDING). L’appel à ServerSupportFunction (HSE_REQ_DONE_WITH_SESSION) doit avoir lieu dans les 30 secondes. Si la fonction TerminateExtension est définie, elle est toujours appelée avec la valeur HSE_TERM_MUST_UNLOAD.

Dans le cas des CGI WebStar, 4D autorise l’envoi fractionné des réponses supérieures à 32 Ko.

4D et WebSTAR (MacOS uniquement)

WebSTAR® est un des serveurs Web les plus répandus sur la plate-forme MacOS. La version 6.7 de 4e Dimension offre la possibilité d’accéder directement à WebSTAR depuis 4D. Inversement, 4D peut répondre directement à une requête de WebSTAR.

Aucun composant supplémentaire n’est requis — en fait, 4D se comporte comme un CGI pour WebSTAR.

Pour configurer cette solution, il suffit de placer un 4D ou un alias de 4D dans le dossier contenant les pages à publier, ou dans le dossier cgi-bin de WebSTAR. Le nom de l’application 4D doit se terminer par “.acgi”, par exemple 4D.acgi. Le serveur Web 4D doit être activé préalablement à toute requête et doit répondre sur un port différent de celui de WebSTAR.

Le principe de fonctionnement est le suivant : lorsqu’un URL ou une action du type “/cgi-bin/4d.acgi$/4daction/proc” est reçue par le serveur WebSTAR, celui-ci envoie un événement spécifique à 4D.acgi via un Apple Event. 4D traite alors l’Apple Event comme une requête HTTP. Dans notre exemple, la requête HTTP serait “/4daction/proc”.

Le mode CGI 4D WebSTAR n’est pas compatible avec le mode contextuel.

Note : Pour plus d'informations sur les possibilités d'interaction entre 4D et WebSTAR, veuillez consulter la documentation de WebSTAR.

Interroger le serveur Web 4D via des CGI (Windows uniquement)

4e Dimension, à compter de la version 6.7, est accompagné de deux extensions, 4DISAPI.DLL et NPH-CGI4D.EXE. Le but de ces extensions est de permettre à un serveur HTTP de transmettre des requêtes au serveur Web 4D. Avec ces extensions, 4D peut être interrogé par tout autre serveur HTTP. Ce mécanisme autorise par exemple le développement de systèmes dans lesquels un serveur Web 4D non sécurisé peut être interrogé via un autre serveur HTTP, tournant lui en mode sécurisé.

Note : Ces deux extensions sont disponibles sous Windows uniquement.

• L'extension 4DISAPI.DLL respecte les spécifications définies par ISAPI (*Internet Services Application Programming Interface*). La technologie ISAPI a été développée à l'origine par Microsoft® pour le serveur IIS, mais a été depuis rendue compatible avec de nombreux serveurs HTTP tels que Netscape®, Apache® ou Sambar®.

• L'extension NPH-CGI4D.EXE respecte les spécifications des CGI (*Common Gateway Interface*) et peut être utilisée avec l'ensemble des serveurs compatibles CGI.

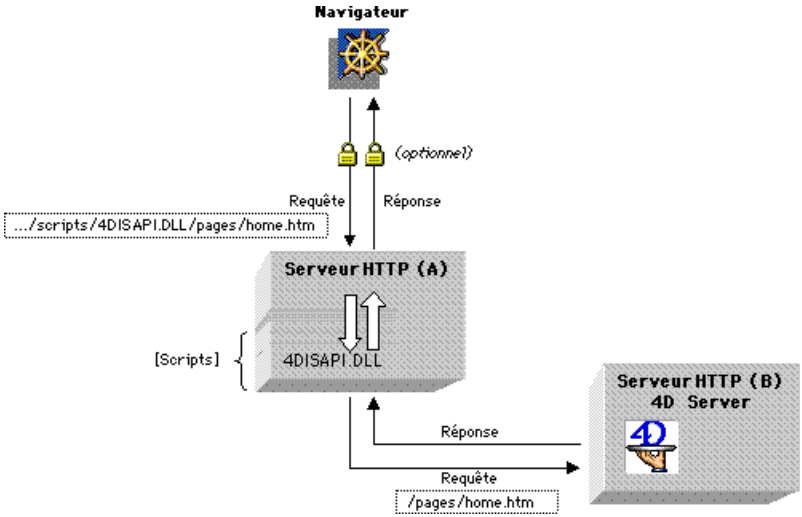
Le fonctionnement de ces deux extensions est identique. La compatibilité CGI est plus largement répandue parmi les serveurs HTTP, toutefois les performances des extensions CGI sont généralement inférieures à celles des extensions ISAPI.

Principe de fonctionnement

Le principe de fonctionnement de ces extensions est le suivant : un serveur HTTP "A" publie des pages sur Internet, et un autre serveur HTTP, "B", est par exemple un 4D Server utilisé en Intranet. Afin que les deux serveurs puissent communiquer, il vous suffit d'ajouter une extension 4DISAPI ou NPH-4DCGI dans le répertoire [Scripts] du serveur A. Lorsqu'un navigateur Web envoie une requête au serveur A, celui-ci la retransmet au serveur B via l'extension 4D ISAPI ou NPH-4DCGI, par l'intermédiaire de l'URL. La réponse est ensuite acheminée au navigateur en sens inverse. Le corps de la requête ou de la réponse HTTP n'est jamais modifié par les extensions.
La requête initiale envoyée au serveur A peut être effectuée en clair ou en mode sécurisé (SSL). La communication entre les deux serveurs HTTP et l'extension 4DISAPI.DLL s'effectue en clair.

Note : Les extensions 4DISAPI et NPH-4DCGI ne sont pas compatibles avec le mode contextuel du serveur Web 4D.

Le schéma suivant illustre ce principe :



• Les extensions reconnaissent les méthodes GET, HEAD et POST, elles gèrent les différents statuts renvoyés par 4D (*200 OK*, *302 Moved Temporarely*, *404 Not Found*...).
• Il n'est pas possible de procéder à une authentification au niveau HTTP via l'extension 4DISAPI ou NPH-CGI4D. Pour cela, il est nécessaire d'utiliser un formulaire HTML (ce qui est parfaitement envisageable en connexion sécurisée).

Note : Les extensions sont avant tout destinées à être utilisées pour recevoir et renvoyer des données dynamiques, et en particulier pour poster des données. Le simple service de pages Web n'offre pas des performances optimales en cas de transit via des extensions ISAPI ou CGI.

Installation et configuration

L'installation des extensions 4DISAPI et NPH-CGI4D s'effectue par simple copie des fichiers 4DISAPI.DLL ou NPH-CGI4D.EXE dans le dossier [Scripts] du serveur HTTP.

Chaque extension installée doit être accompagnée d'un fichier de configuration (de type *.INI). Le fichier .INI doit porter le même nom que l'extension (par exemple 4DISAPI.INI). L'extension et son fichier de configuration doivent être placés dans le même dossier.

Vous pouvez configurer un serveur HTTP de manière à ce qu'il puisse cibler plusieurs autres serveurs HTTP. Dans ce cas, placez dans le dossier [Scripts] du serveur HTTP autant d'extensions que de serveurs-cibles. Il vous suffit de les renommer (par exemple 4DISAPI2.DLL, 4DISAPI3.DLL, etc.). Veillez à insérer un fichier de configuration par extension, et à le renommer en conséquence (4DISAPI2.INI, 4DISAPI3.INI, etc.).

Le fichier .INI est constitué d'une seule section : [Forward]. Cette section accepte les commandes suivantes :

• TargetServer =
Nom ou adresse IP du serveur Web à cibler (par exemple monserveur.net ou 192.193.194.195). Laisser la chaîne vide pour repérer le serveur par son adresse si la résolution par nom n'est pas disponible
Le nom "localhost" est reconnu comme étant l'adresse 127.0.0.1.
Par défaut, l'adresse 127.0.0.1 est utilisée.

• TargetPort =
Port sur lequel le serveur-cible écoute (par exemple 81). Par défaut, le port 8080 est utilisé.

• Timeout =
Délai maximum d'attente de la réponse du serveur (exprimé en secondes). La valeur par défaut est de 30 secondes.

• Allowed =
Liste des URL autorisés, séparés par des virgules. Par exemple : /pages, /img pour n'autoriser que les URL commençant par /pages et /img. Pour autoriser la totalité du site, inscrivez une barre oblique seule / (paramétrage par défaut).

• Forbidden =
Liste des URL interdits, séparés par des virgules. Par exemple : /4dmethod, /pages2 pour interdire les URL commençant par /4dmethod et /pages2. Pour ne rien interdire, ne saisissez rien dans la liste (paramétrage par défaut).
Compte tenu des exemples d'autorisations et d'exclusions fournis ci-dessus, les URL suivants du serveur-cible seront accessibles ou non :

/pages/document.html	accessible
/pages1/onepage.html	accessible
/www/image.gif	inaccessible
/pages2/mypage.html	inaccessible
/4dmethod/myproc	inaccessible

Si un URL déclaré interdit est sollicité, l'extension retourne directement l'erreur "*HTTP/1.0 403 Forbidden*".

Utilisation

Les extensions 4DISAPI et NPH-CGI4D acceptent les URLs suivants :

• **Appel de 4D** (4D ne reçoit que la portion de chemin qui suit le nom de l'extension) :
http://adresse-serveur/cgi-bin/4disapi.dll/[chemin d'accès]
http://adresse-serveur/cgi-bin/nph-cgi4d.exe/[chemin d'accès]

• **Test de fonctionnement de l'extension** (écho de la requête) :
http://adresse-serveur/cgi-bin/4disapi.dll/~~echo
http://adresse-serveur/cgi-bin/nph-cgi4d.exe/~~echo

• **Informations sur l'extension** (support technique) :
http://adresse-serveur/cgi-bin/4disapi.dll/~~info
http://adresse-serveur/cgi-bin/nph-cgi4d.exe/~~info
Les informations suivantes sont retournées :
- nom et version de l'extension, par exemple "Script name: 4disapi.dll (6.7.0b1.2)"
- nom et version du serveur appelant l'extension, par exemple "Server software: 4D_WebStar_D/6.7"
- version du protocole HTTP, par exemple "Server protocol: HTTP/1.0"
- version du protocole CGI, par exemple "Gateway interface: CGI/1.1"

• **Test du serveur cible** (est-il joignable ?) :
http://adresse-serveur/cgi-bin/4disapi.dll/~~target
http://adresse-serveur/cgi-bin/nph-cgi4d.exe/~~target
La réponse est soit :
"Good: target server reached." : le serveur cible a répondu (quel que soit le contenu de la réponse).
soit "Bad: target server not reached." : le serveur est injoignable ou n'a pas répondu.
Dans ce cas, la raison de l'échec peut être :
- pas de fichier de configuration ;
- l'adresse ou le port cible est incorrect ;
- le serveur cible n'est pas en service ;
- le serveur a traité la requête mais est incapable de répondre.

LANCER SERVEUR WEB

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande LANCER SERVEUR WEB démarre la publication de la base de données sur votre réseau Intranet ou sur Internet en utilisant les fonctions de serveur Web internes de 4e Dimension.

Si le serveur Web a été correctement lancé, la variable système OK prend la valeur 1, sinon — si par exemple le composant réseau TCP/IP n'est pas présent — OK prend la valeur 0 (zéro).

Référence

ARRETER SERVEUR WEB.

Ensembles et variables système

Si les services Web sont correctement démarrés, OK prend la valeur 1, sinon OK prend la valeur 0 (zéro).

ARRETER SERVEUR WEB

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande ARRETER SERVEUR WEB stoppe la publication de la base en tant que serveur Web.
Si la base était publiée comme serveur Web, toutes les connexions Web sont interrompues et tous les process Web sont arrêtés.
Si la base n'était pas publiée comme serveur Web, la commande ne fait rien.

Référence

LANCER SERVEUR WEB.

FIXER TEMPORISATION WEB

Serveur Web

version 6.5 (Modifiée)

FIXER TEMPORISATION WEB (timeout)

Paramètre	Type		Description
timeout	Numérique	→	Délai de fermeture de la connexion Web exprimé en secondes

Description

La commande FIXER TEMPORISATION WEB définit le timeout (délai d'attente maximum avant la déconnexion automatique du client Web) pour les process de connexion Web. La valeur du timeout par défaut est de 5 minutes.

Vous pouvez augmenter ou réduire cette valeur en passant dans le paramètre timeout une nouvelle valeur, exprimée en secondes.

La commande prend effet immédiatement et a pour portée la session de travail.
Si la commande FIXER TEMPORISATION WEB est appelée depuis un process Web, la valeur de timeout ne s'applique qu'à ce process.
Si la commande est appelée en-dehors d'un process Web, la valeur de timeout s'applique à tous les process Web créés par la suite.

Référence

Services Web, Process de connexion Web.

FIXER RACINE HTML (chemAccèsHTML)

Paramètre	Type		Description
chemAccèsHTML	Alpha	→	Chemin d'accès HTML au répertoire par défaut des fichiers HTML

Description

La commande FIXER RACINE HTML permet de définir le nouveau répertoire par défaut dans lequel 4D ira rechercher les fichiers HTML passés comme paramètres à ENVOYER FICHIER HTML.
Par défaut, 4D recherche les documents HTML dans le répertoire où se trouve le fichier de structure de la base.

Important : La commande FIXER RACINE HTML ne fonctionne qu'en mode contextuel. Si vous souhaitez définir un répertoire racine spécifique en mode sans contexte, vous devez utiliser l'option "Dossier racine HTML par défaut" des Propriétés de la base.
Il est généralement plus judicieux, pour des raisons de performances, de définir le répertoire racine HTML dans la boîte de dialogue des Propriétés de la base, quel que soit le mode du serveur Web.

Le chemin d'accès au nouveau répertoire doit être exprimé en syntaxe HTML, c'est-à-dire que les noms de répertoires ou de dossiers doivent être séparés par une barre oblique ("/"), quelle que soit la plate-forme que vous utilisez. Pour plus d'informations sur les chemins d'accès et la syntaxe HTML, nous vous conseillons de vous reporter à la partie Programmation de tout manuel sur le langage HTML disponible en librairie.

Si vous passez un chemin d'accès invalide, une erreur liée à la gestion de fichiers du système d'exploitation est générée. Vous pouvez intercepter l'erreur à l'aide d'une méthode installée par la commande APPELER SUR ERREUR. Si la méthode affiche une boîte de dialogue d'alerte ou de message, celle-ci apparaît sur l'écran du browser.

Note : La commande FIXER RACINE HTML tient compte du dossier racine HTML par défaut lorsqu'il est défini dans les Propriétés de la base.
Pour plus d'informations, reportez-vous à la section Services Web, Sécurité des connexions.

Exemple

Reportez-vous à l'exemple de la commande ENVOYER FICHIER HTML.

Référence

APPELER SUR ERREUR.

FIXER LIMITES AFFICHAGE WEB

Serveur Web

version 6.0

FIXER LIMITES AFFICHAGE WEB (nombreEnr{; nombrePages{; imageRéf}})

Paramètre	Type		Description
nombreEnr	Numérique	→	Nombre maximum d'enregistrements à afficher dans chaque page HTML
nombrePages	Numérique	→	Nombre maximum de références de pages en bas de chaque page HTML
imageRéf	Numérique	→	Numéro de référence d'image pour l'icône du bouton d'affichage pleine page d'un enregistrement

Description

La commande FIXER LIMITES AFFICHAGE WEB modifie la manière dont 4e Dimension affiche une sélection d'enregistrements dans un browser Web lorsque vous utilisez les commandes VISUALISER SELECTION ou MODIFIER SELECTION.

Lorsque vous affichez une sélection d'enregistrements avec 4e Dimension ou 4D Client, le programme ne charge pas tous les enregistrements de la sélection, mais uniquement ceux qui sont visibles dans la fenêtre. Ainsi, même si des milliers d'enregistrements sont sélectionnés, leur affichage reste rapide : seuls les enregistrements visibles simultanément sont chargés du disque. Ensuite, si vous faites défiler la liste d'enregistrements ou redimensionnez la fenêtre, 4D charge en conséquence les enregistrements.

Avec le Web, 4D découpe en pages la sélection d'enregistrements à afficher. En fait, sans système de pagination, l'affichage d'une sélection de milliers d'enregistrements signifierait la circulation de milliers d'enregistrements sur votre réseau Intranet ou sur Internet et leur affichage dans une seule page Web. Non seulement cela prendrait un certain temps pour télécharger tous les enregistrements mais, de plus, votre browser Web serait rapidement à court de mémoire.

Par défaut, 4e Dimension affiche les 20 premiers enregistrements (s'ils existent) de la sélection et insère à la fin de chaque page HTML 20 liens vers les 20 premières pages (si elles existent) de la sélection. Cela signifie que par défaut, vous pouvez accéder aux 400 premiers enregistrements (s'ils existent) de la sélection en cliquant sur le lien de page placé à la fin de chaque page de la sélection. Notez que ce système de pagination ne nécessite aucune ligne de code, il se met automatiquement en place lors de l'exécution des commandes VISUALISER SELECTION ou MODIFIER SELECTION.

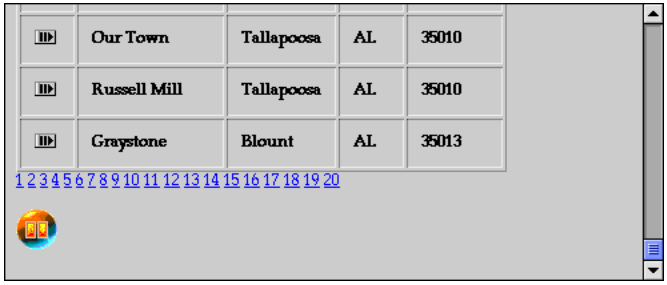
FIXER LIMITES AFFICHAGE WEB vous permet de modifier les paramétrages. Dans nombreEnr, vous indiquez le nombre maximum d'enregistrements qui doivent être affichés par page de la sélection. Dans nombrePages, vous indiquez le nombre maximum de liens vers d'autres pages de la sélection qui doivent être affichés en bas de chaque page de la sélection.

Si, par exemple, vous disposez d'une sélection de 10 000 enregistrements et voulez absolument pouvoir naviguer parmi la totalité de ces enregistrements dans une seule sélection, vous pouvez passer 100 dans les paramètres nombreEnr et nombrePages. Rappelez-vous toutefois que les données circulant sur le réseau ou sur Internet, avec ce dernier la vitesse est un facteur important à prendre en considération lorsque vous modifiez les paramètres d'affichage de la sélection.

Enfin, FIXER LIMITES AFFICHAGE WEB vous permet de remplacer l'icône par défaut du bouton d'affichage d'un enregistrement en pleine page. Pour cela, passez dans le paramètre imageRéf le numéro de référence de l'image, stockée dans la librairie d'images de la base, que vous souhaitez utiliser comme nouvelle icône.
FIXER LIMITES AFFICHAGE WEB n'affecte que les appels ultérieurs aux commandes VISUALISER SELECTION et MODIFIER SELECTION et sa portée est locale au process courant.

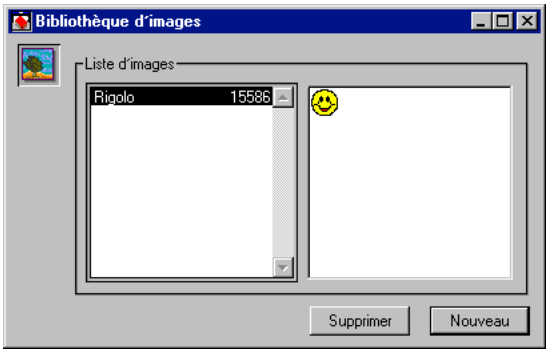
Exemple

Dans l'exemple suivant, un VISUALISER SELECTION ou un MODIFIER SELECTION est exécuté pour la table [Codes américains]. Au niveau du browser Web, 4D affiche par défaut les enregistrements de la manière suivante :



Notez que vous pouvez naviguer parmi les 400 premiers enregistrements.

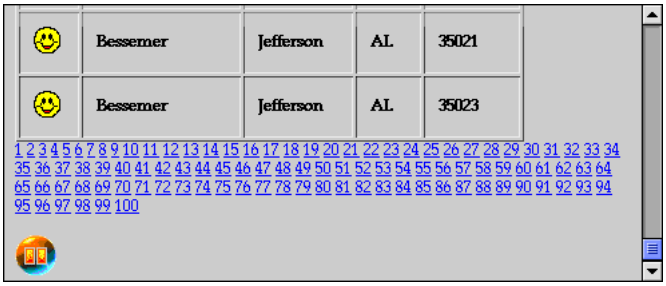
Si l'image suivante est placée dans la bibliothèque d'images de la base :



Si, également, la méthode projet qui affiche la sélection exécute l'instruction suivante avant d'appeler VISUALISER SELECTION ou MODIFIER SELECTION :

⇒ **FIXER LIMITES AFFICHAGE WEB** (50;100;15586)

Alors, au niveau du browser Web, la sélection des enregistrements apparaîtra ainsi :



Vous pouvez donc naviguer parmi les 5 000 premiers enregistrements de la sélection.

Référence

MODIFIER SELECTION, VISUALISER SELECTION.

FIXER PAGE ACCUEIL (homePage)

Paramètre	Type		Description
homePage	Alpha	→	Nom de page ou chemin d'accès HTML à la page ou "" pour ne pas envoyer de page d'accueil personnalisée

Description

La commande FIXER PAGE ACCUEIL vous permet de définir une page d'accueil (page Home) personnalisée pour le process Web courant. Par défaut, en mode contextuel, la barre de menus courante est appelée en tant que page d'accueil.

La page définie est liée au process Web, vous pouvez donc définir des pages d'accueil différentes en fonction, par exemple, de l'utilisateur connecté. Cette page peut être statique ou semi-dynamique.

Vous passez dans le paramètre homePage le nom de la page HTML d'accueil ou le chemin d'accès HTML complet à la page.

Pour ne plus envoyer homePage comme page d'accueil pour le process Web courant, appelez de nouveau la commande FIXER PAGE ACCUEIL en passant une chaîne vide ("") dans homePage.

Note : Vous pouvez également définir une page d'accueil par défaut dans les Propriétés de la base. Dans ce cas, la page s'applique par défaut à toutes les connexions Web, quel que soit le mode de démarrage (contextuel ou sans contexte) du serveur Web.

ENVOYER FICHIER HTML (fichierHTML)

Paramètre	Type		Description
fichierHTML	Alpha	→	Chemin d'accès HTML au fichier HTML ou Chaîne vide pour terminer ENVOYER FICHIER HTML

Description

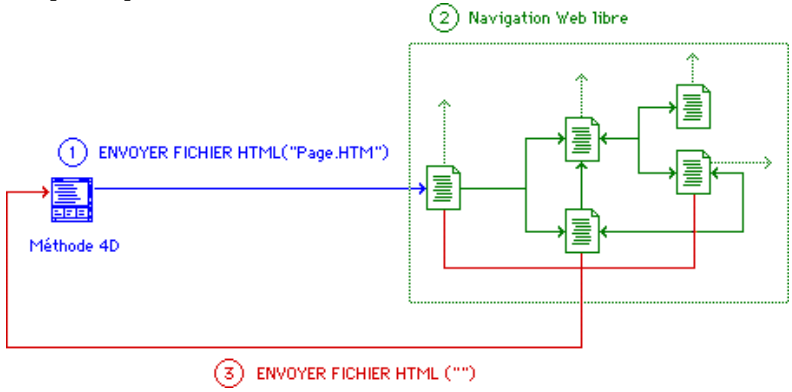
La commande ENVOYER FICHIER HTML envoie au browser Web la page Web stockée dans le document HTML dont vous passez le chemin d'accès dans fichierHTML.

Par défaut, 4e Dimension recherche le document HTML à l'intérieur du répertoire ou dossier dans lequel se trouve le fichier de structure de la base, à moins que vous n'ayez défini l'emplacement par défaut des documents HTML dans un autre répertoire ou dossier dans la boît de dialogue des Propriétés de la base.

Si vous passez un chemin d'accès HTML invalide, une erreur liée à la gestion de fichiers de votre système d'exploitation est générée. Vous pouvez intercepter l'erreur à l'aide d'une méthode installée par la commande APPELER SUR ERREUR. Si la méthode affiche une boîte de dialogue d'alerte ou de message, celle-ci apparaît sur le poste du browser.

La syntaxe ENVOYER FICHIER HTML("") (vous passez une chaîne vide dans le paramètre fichierHTML) permet d'achever l'appel à la commande ENVOYER FICHIER HTML ayant démarré le mode HTML.

Ce principe est illustré dans le schéma suivant :



(1) Une méthode 4D (projet, objet ou base) exécute un ENVOYER FICHIER HTML, envoyant un document HTML au browser.

(2) La page Web initiale envoyée au browser peut comporter des liens HTML vers d'autres pages Web ou peut référencer elle-même des méthodes 4D appelant ENVOYER FICHIER HTML pour envoyer d'autres pages Web. Ces autres pages peuvent également comporter des liens ou référencer des méthodes 4D pour permettre d'accéder à d'autres pages, etc. Pendant que vous naviguez parmi les pages Web, vous pouvez également utiliser les outils de navigation du browser, comme le bouton Précédent.

(3) N'importe laquelle des pages Web peut contenir une référence à une méthode 4D qui exécute l'instruction ENVOYER FICHIER HTML(""). Cette instruction termine l'appel à ENVOYER FICHIER HTML qui a lancé le processus et permet de retourner à l'exécution de la méthode 4D ayant initialement démarré la navigation Web libre.

Une fois que l'instruction ENVOYER FICHIER HTML a été exécutée, la variable système *OK* est mise à jour : si le fichier à envoyer existe et si le *timeout* n'est pas dépassé, la variable *OK* prend la valeur 1. Dans le cas contraire, elle prend la valeur 0.

Note : Si vous appelez ENVOYER FICHIER HTML depuis un process qui n'est pas un process de connexion Web, la commande ne fait rien. Aucune erreur n'est retournée, l'appel est simplement ignoré.

Les éventuelles références aux variables 4D et aux balises 4DSCRIPT dans la page sont toujours analysées, quel que soit le mode.

Exemples

(1) Un document HTML intitulé "HomePage.HTM" est placé dans le même répertoire que le fichier de structure de la base. Ce document est une page Web que vous voulez envoyer à tout utilisateur Web se connectant à votre base, au lieu de la barre de menus par défaut n°1. Pour cela, écrivez dans la méthode base Sur connexion Web de votre application :

```
` Méthode base Sur connexion Web
⇒ ENVOYER FICHIER HTML ("HomePage.HTM")
```

(2) Le répertoire de la base est organisé de la manière suivante :

```
..\Documents\EnCours\Bases\MaBase.4DB
..\Documents\EnCours\Bases\MaBase.RSR
..\Documents\EnCours\Bases\MaBase.4DD
..\Documents\EnCours\Bases\WebDocs\HTM\HomePage.HTM
```

L'envoi de la page Web "HomePage.HTM" peut être effectué de cette manière :

```
⇒ ENVOYER FICHIER HTML ("WebDocs\HTM\HomePage.HTM")
```

ou bien de cette manière :

```
FIXER RACINE HTML ("WebDocs/HTM/")
⇒ ENVOYER FICHIER HTML ("HomePage.HTM")
```

(3) Pendant une session Web 4D, vous ajoutez des enregistrements à l'aide d'un formulaire 4D. Dans ce formulaire se trouve un bouton bAide dont la méthode objet est la suivante :

```
` Méthode objet du bouton bAide
⇒ ENVOYER FICHIER HTML ("Aide.HTM")
```

A partir du document Aide.HTM, vous pouvez librement naviguer parmi différentes pages HTML qui implémentent un système d'aide en ligne pour votre site Web. Dans chaque page, vous placez un bouton "submit" dont le titre est Fin, permettant de retourner à la saisie de données. Pour cela, chaque document HTML doit comporter cette définition pour le bouton "submit" :

```
<!--bTerminé bouton submit-->
<P><INPUT TYPE="submit" NAME="bFin" VALUE="Done"></P>
```

ainsi que cette définition de l'action FORM POST :

```
<!--Exécution de la méthode 4D htm_Aide_Fin si clic sur un bouton submit-->
<FORM action="/4DMETHOD/htm_Aide_Fin" method="POST">
```

Du côté de 4D, la méthode projet htm_Aide_Fin termine l'appel à ENVOYER FICHIER HTML, initié par le bouton bAide :

```
      ` Méthode projet htm_Aide_Fin
⇒     ENVOYER FICHIER HTML ("")
```

L'appel à ENVOYER FICHIER HTML est la dernière ligne de la méthode objet du bouton bAide ; la méthode s'achève alors et vous retournez à la saisie de données.

Variables et ensembles système

Si le fichier à envoyer existe et si le *timeout* n'est pas dépassé, la variable *OK* prend la valeur 1. Dans le cas contraire, elle prend la valeur 0.

Référence

Encapsulation HTML et Javascript, ENVOYER BLOB HTML, Services Web, Premiers pas (Partie II), Services Web, Premiers pas (Partie II).

ENVOYER BLOB HTML

Serveur Web

version 6.5

ENVOYER BLOB HTML (blob; type{; sansContexte})

Paramètre	Type		Description
blob	BLOB	→	BLOB à envoyer au browser
type	Alpha	→	Type de données du BLOB
sansContexte	Booléen	→	Vrai = Passer en mode sans contexte, Faux ou omis = Conserver mode courant

Description

La commande ENVOYER BLOB HTML permet d'envoyer le BLOB blob au browser.

Le type de données contenues dans le BLOB est indiqué par le paramètre type. Ce paramètre peut contenir les valeurs suivantes :
• type = **Chaîne vide** ("") : dans ce cas, vous ne fournissez aucune information sur le BLOB. Le browser tentera alors d'interpréter lui-même le contenu du BLOB.
• type = **Extension de fichier** (ex. : ".HTM", ".GIF", ".JPEG", etc.) : dans ce cas, vous fournissez au browser, par l'intermédiaire de son extension, le type MIME des données contenues dans le BLOB. Le BLOB sera interprété en fonction de cette extension. Toutefois, l'extension doit être standard afin que le browser puisse l'interpréter correctement. Une liste des types MIME les plus courants et de leurs extensions est fournie ci-dessous.
• type = **Mime/Type** (ex. : "text/html", "image/tiff", etc.) : dans ce cas, vous fournissez directement au browser le type MIME des données contenues dans le BLOB. Cette solution est celle qui vous offre le plus de latitude. En effet, outre les types standard, vous pouvez passer un type MIME personnalisé pour envoyer des documents propriétaires en Intranet. Il vous suffit pour cela de configurer les browsers afin qu'ils reconnaissent le type envoyé et, par exemple, exécutent l'application correspondante. La valeur à passer dans le paramètre type est, dans ce cas "application/x-[NomDuType]"). Dans les browsers des postes clients, vous référencez ce type et lui associez l'action "Exécuter l'application". La commande ENVOYER BLOB HTML vous permet alors d'envoyer des documents de tout type, les clients Intranet ouvrant automatiquement l'application associée.

Note : Si le BLOB est de type "text/html" (.htm, .html, .shtm, .shtml), il est traduit et analysé comme un fichier HTML. Dans ce cas, lorsqu'elle est utilisée en mode sans contexte, ENVOYER BLOB HTML fonctionne exactement comme ENVOYER FICHIER HTML. En particulier, une référence à une méthode 4D qui exécute l'instruction ENVOYER BLOB HTML("") doit être présente dans les pages reçues par le browser, afin de terminer l'appel à ENVOYER BLOB HTML et de retourner à l'exécution de la méthode 4D d'appel. Pour plus d'informations sur ce mécanisme, reportez-vous à la description de la commande ENVOYER FICHIER HTML.

Voici une liste des types MIME les plus courants :

Extension	Mime/Type
.htm	text/html
.html	text/html
.shtml	text/html
.shtm	text/html
.css	text/css
.pdf	application/pdf
.rtf	application/rtf
.ps	application/postscript
.eps	application/postscript
.hqx	application/mac-binhex40
.js	application/javascript
.txt	text/plain
.text	text/plain
.gif	image/gif
.jpg	image/jpeg
.jpeg	image/jpeg
.jpe	image/jpeg
.jfif	image/jpeg
.pic	image/pict
.pict	image/pict
.tif	image/tiff
.tiff	image/tiff
.mpeg	video/mpeg
.mpg	video/mpeg
.mov	video/quicktime
.moov	video/quicktime
.aif	audio/aiff
.aiff	audio/aiff
.wav	audio/wav
.ram	audio/x-pn-realaudio
.sit	application/x-stuffit
.bin	application/x-stuffit
.z	application/x-zip
.zip	application/x-zip
.gz	application/x-gzip
.tar	application/x-tar

Note : Pour plus d'informations sur ce point, veuillez consulter la section “Protocol Numbers and Assignment Services” sur le site http://www.iana.org.

Le paramètre sansContexte vous permet d’indiquer au serveur Web 4D que vous souhaitez passer du mode contextuel au mode sans contexte.
Pour utiliser le mode sans contexte, passez Vrai dans le paramètre sansContexte.
Si ce paramètre est omis ou s’il contient Faux, l'envoi du BLOB est effectué dans le mode courant.

Les éventuelles références aux variables 4D et balises 4DSCRIPT dans la page sont toujours analysées, quel que soit le mode.

Exemple

Reportez-vous à l'exemple de la routine IMAGE VERS GIF.

Référence

ENVOYER FICHIER HTML, Services Web, Mode sans contexte.

ENVOYER TEXTE HTML (texteHTML{; sansContexte})

Paramètre	Type		Description
texteHTML	Texte	→	Champ ou variable texte au format HTML à envoyer au browser
sansContexte	Booléen	→	Vrai = Passage en mode sans contexte Faux ou omis = Conservation du mode en cours

Description

La commande ENVOYER TEXTE HTML permet d'envoyer directement des données texte formatées en HTML.

Le paramètre texteHTML contient les données à envoyer. 4D n'effectue aucun contrôle sur le contenu de ce paramètre, vous devez donc veiller à ce que le codage HTML soit correct. Le texte doit être encodé en ISO Latin-1.

Note : Cette commande équivaut strictement à l'envoi d'un BLOB ayant le type "text/html" à l'aide de la commande ENVOYER BLOB HTML.

Le paramètre sansContexte vous permet d'indiquer au serveur Web 4D que vous souhaitez passer du mode "contextuel" au mode "sans contexte" lors de l'exécution de cette commande.
Pour utiliser le mode sans contexte, passez Vrai dans sansContexte. Pour conserver le mode courant, passez Faux dans sansContexte ou omettez ce paramètre.

Les éventuelles références aux variables 4D et balises 4DSCRIPT dans le texte sont toujours analysées, quel que soit le mode.

Exemple

La méthode suivante :

```
TEXTE VERS BLOB("<html><head></head><body>"+Chaine(Heure courante)+
                                    "</body></html>";$blob;Texte sans longueur)
ENVOYER BLOB HTML ($blob;"text/html")
```

... peut être remplacée par :

```
⇒ ENVOYER TEXTE HTML ("<html><head></head><body>"+Chaine(Heure courante)+
                                                    "</body></html>")
```

Référence

ENVOYER BLOB HTML, Mac vers ISO.

LIRE VARIABLES FORMULAIRE WEB (tabNoms; tabValeurs)

Paramètre	Type		Description
tabNoms	Tableau Texte	←	Noms des variables du formulaire Web
tabValeurs	Tableau Texte	←	Valeurs des variables du formulaire Web

Description

La commande LIRE VARIABLES FORMULAIRE WEB remplit les tableaux texte tabNoms et tabValeurs avec, respectivement, les noms et les valeurs des variables contenues dans un formulaire Web "soumis" (c'est-à-dire envoyé au serveur Web).
Cette commande récupère la valeur de toutes les variables pouvant être incluses dans des pages HTML : zones de saisie, boutons, cases à cocher, boutons radio, pop up menus, listes d'options...

Note: Dans le cas des cases à cocher, le nom de la variable et sa valeur ("ON") ne sont retournés que si la case est effectivement cochée.

La commande fonctionne en mode contextuel et en mode sans contexte, lorsqu'elle est appelée dans les conditions suivantes :
• un formulaire est soumis avec la méthode "POST" (action commençant par /4DACTION, /4DMETHOD ou /4DCGI),
• un formulaire est soumis avec la méthode "GET" (action commençant par /4DACTION, /4DMETHOD ou /4DCGI),
• un URL contenant une chaîne d'interrogation est envoyé au serveur Web.

Cette commande peut être appelée, selon les besoins, dans la Méthode base Sur connexion Web ou toute autre méthode 4D qui résulte de la soumission d'un formulaire.

Précisions sur les formulaires Web et les actions associées

Un formulaire est composé de "zones de saisie" (zones de texte, boutons, cases à cocher), chacune ayant un nom. Lorsqu'un formulaire est "soumis" (une requête est envoyée au serveur Web), la requête comporte, entre autres, la liste des zones de saisie et leurs valeurs respectives.
Il y a deux "méthodes" pour soumettre un formulaire (4D accepte indifféremment l'une ou l'autre) :
• POST, généralement utilisée pour l'insertion de données dans le serveur Web — vers une base de données,
• GET, généralement utilisée pour l'interrogation du serveur Web — données en provenance d'une base de données.

Exemple

Un formulaire contient deux champs, vNOM et vVILLE, qui reçoivent les valeurs “MARTIN” et “PARIS”. L’action associée au formulaire est “/4DACTION/WEBFORM”.
• Si la méthode du formulaire est POST (cas le plus souvent utilisé), les données saisies ne seront pas visibles dans l’URL (c’est-à-dire http://127.0.0.1/4DACTION/WEBFORM).
• Si la méthode du formulaire est GET, les données seront visibles dans l’URL (c’est-à-dire http://127.0.0.1/4DACTION/WEBFORM?vNOM=MARTIN&vVILLE=PARIS).

La méthode *WEBFORM* peut être de la forme suivante :

```
   TABLEAU TEXTE($tnoms;0)
   TABLEAU TEXTE($tvaleurs;0)
⇒  LIRE VARIABLES FORMULAIRE WEB($tnoms;$tvaleurs)
```

On obtient alors :

```
$tnoms{1} = "vNOM"
$tnoms{2} = "vVILLE"
$tvaleurs{1} = "MARTIN"
$tvaleurs{2} = "PARIS"
```

Référence

Encapsulation HTML et Javascript, Services Web, URLs et actions de formulaires.

Contexte Web

Serveur Web

version 6.5

Contexte Web → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Vrai = Mode contextuel Faux = Mode sans contexte

Description

La commande Contexte Web doit être appelée depuis un process Web. Elle retourne un booléen indiquant si la connexion Web s'effectue (Vrai) ou non (Faux) en mode contextuel.

Note : Appelée depuis un process autre qu'un process Web, cette fonction retourne toujours Faux.

L'utilisation de cette fonction est préconisée dans la Méthode base Sur connexion Web.

Exemple

Voici la structure typique de la Méthode base Sur connexion Web :

```
⇒   Si (Contexte Web)
       AvecContexte ($1;$2;$3;$4;$5;$6)
    Sinon
       SansContexte ($1;$2;$3;$4;$5;$6)
    Fin de si
```

Référence

INFORMATIONS PROCESS, Méthode base Sur connexion Web, Services Web, Mode sans contexte, Services Web, Process de connexion Web.

FIXER ENTETE HTTP (entête|tabChamps{; tabValeurs})

Paramètre	Type		Description
entête | tabChamps	Texte | Tab Texte	→	En-tête HTTP de la requête ou Champs de l'en-tête HTTP
tabValeurs	Tab Texte	→	Contenu des champs de l'en-tête HTTP

Description

La commande FIXER ENTETE HTTP permet de fixer les champs de l'en-tête HTTP de la réponse faite au browser Web par 4D. Elle n'a d'effet que dans un process Web en mode sans contexte.
Cette commande vous permet, en particulier, de gérer des "cookies".

FIXER ENTETE HTTP admet deux syntaxes :

• **Première syntaxe : FIXER ENTETE HTTP (entête)**
Vous passez dans le paramètre entête, de type texte, les champs de l'en-tête HTTP que vous souhaitez fixer. Les champs doivent être séparés entre eux par un retour chariot ou une séquence cr/lf (retour chariot/retour à la ligne), sous Windows et MacOS, 4D se charge du formatage de la réponse.
Exemple : "HTTP/1.0 200 OK"+Caractere(13)+"Set-Cookie: C=HELLO".
Pour plus d'informations sur la syntaxe à appliquer, veuillez consulter sur Internet les R.F.C (Request For Comments) à l'adresse http://www.w3c.org.

• **Deuxième syntaxe : FIXER ENTETE HTTP (tabChamps; tabValeurs)**
L'en-tête HTTP est défini à l'aide de deux tableaux texte, tabChamps et tabValeurs.
L'en-tête sera écrit de la manière suivante :

```
tabChamps{1} = "X-VERSION"      tabValeurs{1} = "HTTP/1.0" *
tabChamps{2} = "X-STATUS"       tabValeurs{2} = "200 OK" *
tabChamps{3} = "Set-Cookie"     tabValeurs{3} = "C=HELLO"
```

* Ces deux premiers éléments constituent la première ligne de la réponse. Lorsqu'ils sont saisis, ils doivent impérativement être les éléments 1 et 2 des tableaux. Il est toutefois possible de les omettre et d'écrire seulement — 4D se chargeant de formater l'en-tête :

```
tabChamps{1} = "Set-Cookie"     tabValeurs{1} = "C=HELLO"
```

Si vous ne spécifiez pas de statut, celui-ci est automatiquement HTTP/1.0 200 OK.
Conformément à la norme HTTP, les noms des champs HTTP sont toujours libellés en anglais.

Note : Si plusieurs appels à FIXER ENTETE HTTP ont lieu dans le même process Web, seul le dernier appel est pris en compte.

Les champs **Content-Length**, **Server** et **Date** sont toujours fixés par 4D.

Exemple

Voici un exemple de “cookie” personnalisé :

```
⇒ FIXER ENTETE HTTP("SET-COOKIE: USER="+Chaine(Abs(Hasard))+"; PATH=/")
```

Référence

LIRE ENTETE HTTP.

LIRE ENTETE HTTP (entête | tabChamps{; tabValeurs})

Paramètre	Type		Description
entête | tabChamps	Texte | Tab Texte	←	En-tête HTTP de la requête ou Champs de l'en-tête HTTP
tabValeurs	Tab Texte	←	Contenu des champs de l'en-tête HTTP

Description

La commande LIRE ENTETE HTTP retourne, soit sous forme de chaîne, soit sous forme de deux tableaux, l'en-tête HTTP de la requête en cours de traitement.

Cette commande fonctionne uniquement en mode sans contexte. Elle peut être appelée depuis toute méthode (Méthode base Sur connexion Web, Méthode base Sur authentification Web, méthode appelée par "/4DACTION"...) exécutée dans un process Web en mode sans contexte.
Appelée en mode contextuel, LIRE ENTETE HTTP retourne des chaînes vides.

• **Première syntaxe : LIRE ENTETE HTTP (entête)**
Lorsque vous utilisez cette syntaxe, le résultat retourné dans la variable entête est du type suivant :

```
"GET /page.html HTTP\1.0"+Caractere(13)+Caractere(10)+"User-Agent:
browser"+Caractere(13)+Caractere(10)+"Cookie: C=HELLO"
```

Chaque champ d'en-tête est séparé par une séquence CR+LF (*Retour chariot+Retour à la ligne*), sous Windows et MacOS.

• **Seconde syntaxe : LIRE ENTETE HTTP (tabChamps; tabValeurs)**
Lorsque vous utilisez cette syntaxe, les résultats retournés dans les tableaux tabChamps et tabValeurs sont du type suivant :

tabChamps{1} = "X-METHOD"	tabValeurs{1} = "GET" *
tabChamps{2} = "X-URL"	tabValeurs{2} = "/page.html" *
tabChamps{3} = "X-VERSION"	tabValeurs{3} = "HTTP/1.0" *
tabChamps{4} = "User-Agent"	tabValeurs{4} = "browser"
tabChamps{5} = "Cookie"	tabValeurs{5} = "C=HELLO"

* Ces trois premiers éléments ne correspondent pas à des champs HTTP. Ils constituent la première ligne de la requête.

Conformément à la norme HTTP, les noms des champs sont toujours libellés en anglais.

A titre indicatif, voici une liste non exhaustive des champs HTTP pouvant être présents dans une requête :
• Accept : ce que le navigateur est susceptible d'accepter comme contenu.
• Accept-Language : la ou les langue(s) acceptée(s) par le navigateur (pour information). Permet de choisir une page d'accueil en fonction de la langue préférée du navigateur.
• Cookie : liste des cookies.
• From : adresse e-mail de l'utilisateur du navigateur.
• Host : nom ou adresse du serveur (par exemple, dans le cas de l'URL http://monserveurweb/mapage.html, Host prend la valeur "monserveurweb"). Permet de gérer les cas où plusieurs noms pointent vers la même adresse IP (virtual hosting).
• Referer : provenance de la requête (par exemple http://monserveurweb/mapage1.html), c'est-à-dire la page que l'utilisateur affiche s'il clique sur le bouton **Précédent** de son navigateur.
• User-Agent : nom et version du navigateur ou du proxy.

Exemple

• Cette méthode récupère le contenu de tout champ d'en-tête de requête HTTP :

```
      ` Méthode projet GetHTTPField
      ` GetHTTPField ( Texte ) -> Texte
      ` GetHTTPField ( Nom en-tête HTTP ) -> Contenu en-tête HTTP
   C_TEXTE($0;$1)
   C_ENTIER LONG($vlElem)
   TABLEAU TEXTE($noms;0)
   TABLEAU TEXTE($valeurs;0)
   $0:=""
⇒  LIRE ENTETE HTTP ($noms;$valeurs)
   $vlElem:=Chercher dans tableau($noms;$1)
   Si ($vlElem>0)
      $0:=$valeurs{$vlElem}
   Fin de si
```

• Une fois que cette méthode projet est écrite, vous pouvez l'appeler ainsi :

```
      ` Contenu de l'en-tête Cookie
   $cookie:=GetHTTPField("Cookie")
```

• Vous pouvez également envoyer des pages différentes en fonction de la langue du navigateur (par exemple dans la Méthode base Sur connexion Web) :

```
   $langue:=GetHTTPField("Accept-Language")
   Au cas ou
      :($langue="@fr@")    `Français (cf. liste ISO 639)
         ENVOYER FICHIER HTML("index_fr.html")
      :($langue="@es@")    `Espagnol (cf. liste ISO 639)
         ENVOYER FICHIER HTML("index_es.html")
      Sinon
         ENVOYER FICHIER HTML("index.html")
   Fin de cas
```

• Exemple de gestion des hôtes virtuels (par exemple dans la Méthode base Sur connexion Web). Les trois noms “home_site.com”, “home_site1.com” et “home_site2.com” pointent vers la même adresse IP, par exemple 192.1.2.3.

```
$host:=GetHTTPField("Host")
Au cas ou
   :($host="www.site1.com")
      ENVOYER FICHIER HTML("home_site1.com")
   :($host="www.site2.com")
      ENVOYER FICHIER HTML("home_site2.com")
   Sinon
      ENVOYER FICHIER HTML("home_site.com")
Fin de cas
```

Référence

FIXER ENTETE HTTP.

ENVOYER REDIRECTION HTTP (url{; *})

Paramètre	Type		Description
url	Alpha	→	Nouvel URL
*	*	→	Si spécifié = l'URL n'est pas traduit, Si omis = l'URL est traduit

Description

La commande ENVOYER REDIRECTION HTTP permet de transformer un URL en un autre.

Le paramètre url contient le nouvel URL qui permet de rediriger la requête. Si ce paramètre est un url vers un fichier, il doit contenir la référence à ce fichier, comme en mode sans contexte, par exemple : ENVOYER REDIRECTION HTTP ("/MaPage.HTM").

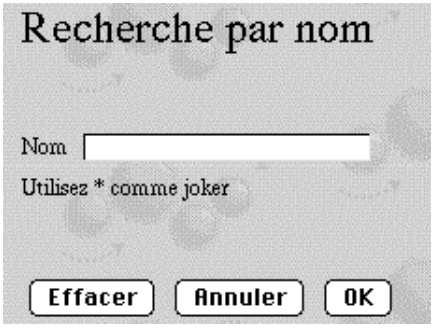
Lorsque cette commande est appelée depuis le mode contextuel, le process Web est tué juste après son exécution. Elle prévaut sur les commandes d'envoi de données (ENVOYER FICHIER HTML, ENVOYER BLOB HTML, etc.) éventuellement placées dans la même méthode.

Cette commande permet également de rediriger une requête vers un autre serveur Web.

4D encode automatiquement les caractères spéciaux de l'URL. Si vous passez le caractère *, 4D ne traduira pas les caractères spéciaux de l'URL.

Exemple

Vous pouvez utiliser cet URL pour effectuer, à l'aide de pages statiques, des recherches personnalisées dans 4D. Imaginez que vous ayez placé dans une page HTML statiques les éléments suivants :



L'action POST "/4dcgi/rech" a été associée à la zone de texte et aux boutons **OK** et **Annuler**.

Dans la partie (ou la sous-méthode) de la Méthode base Sur connexion Web gérant le mode sans contexte, placez les instructions suivantes :

```
      Au cas ou
         : ($1="/4dcgi/rech")   `Lorsque 4D reçoit cet URL
               `Si le bouton OK a été utilisé et le champ 'nom' contient une valeur
            Si ((bOK="OK") & (nom # ""))
                  `Transformer l'URL afin d'exécuter le code de la recherche, placé plus
                  `loin dans la même méthode
⇒                ENVOYER REDIRECTION HTTP("/4dcgi/rech?"+nom)
            Sinon     `Sinon retourner à la page de départ
⇒                ENVOYER REDIRECTION HTTP("/page1.htm")
            Fin de si
            ...

         : ($1="/4dcgi/rech?@")   `Si l'URL a été redirigé
            ... `Placez ici le code de la recherche
      Fin de cas
```

STATISTIQUES DU CACHE WEB

Serveur Web

version 6.5

STATISTIQUES DU CACHE WEB (pages; hits; usage)

Paramètre	Type		Description
pages	Tab Texte	←	Noms des pages les plus consultées
hits	Tab Entier long	←	Nombre de hits pour chaque page
usage	Numérique	←	Pourcentages du cache utilisé

Description

La commande STATISTIQUES DU CACHE WEB vous permet d'obtenir des informations sur les pages les plus consultées, chargées dans le cache du serveur Web. Par conséquent, ces statistiques concernent uniquement les pages statiques, les images GIF, les images JPEG <100 ko et les feuilles de style (.css).

Note : Pour plus d'informations sur le paramétrage du cache du serveur Web 4D, reportez-vous à la section Services Web, Paramétrages du Serveur Web.

La commande remplit le tableau texte pages avec les noms des pages les plus consultées. Le tableau entier long hits reçoit le nombre de "hits" pour chaque page. La variable numérique usage reçoit le pourcentage du cache Web utilisé par chaque page.

Exemple

Vous souhaitez générer une page semi-dynamique affichant les statistiques d'utilisation du cache Web. Pour cela, dans une page HTML statique appelée "stats.shtm", vous placez la balise <!--4DACTION/STATS-->. Vous placez ensuite des références aux variables *vPages* et *vUsage*.
Dans la méthode projet *STATS*, écrivez le code suivant :

```
      C_TEXTE($1)
      TABLEAU TEXTE (pages;0)
      TABLEAU ENTIER LONG (hits;0)
      C_ENTIER LONG (vUsage)
⇒     STATISTIQUES DU CACHE WEB(pages;hits;vUsage)
      vPages:=Caractere(1)
      Boucle ($i;1;Taille tableau(pages))
            `Pour chaque page présente dans le cache
         vPages:=vPages+pages{$i}+"   "+Chaine(hits{$i})+"<br>"
            `Insertion du nom de la page et du code HTML
      Fin de boucle
      ENVOYER FICHIER HTML("stats.shtm")
            `Le contenu des pages suffixées ".shtm" est automatiquement analysé
```

Référence

Services Web, Paramétrages du Serveur Web.

Connexion Web securisee

Connexion Web securisee → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	Vrai = la connexion Web est sécurisée Faux = la connexion Web n'est pas sécurisée

Description

La commande Connexion Web securisee retourne un booléen indiquant si la connexion au serveur Web 4D s'effectue en mode sécurisé via SSL (la requête débute par "https:" au lieu de "http:").

• Si la connexion est effectuée en SSL, la fonction retourne Vrai.
• Si la connexion est effectuée en mode classique (non sécurisé), la fonction retourne Faux.

Cette commande permet par exemple, le cas échéant, de refuser les tentatives de connexion en mode non sécurisé. Pour plus d'informations sur ce point, reportez-vous à la section Services Web, Utiliser le protocole SSL.

Cette commande permet par exemple, le cas échéant, de refuser les tentatives de connexion en mode non sécurisé.

Référence

GENERER DEMANDE CERTIFICAT, Services Web, Utiliser le protocole SSL.

OUVRIR URL WEB (url{; *})

Paramètre	Type		Description
url	Alpha	→	URL de démarrage
*	*	→	Si spécifié = l'URL n'est pas traduit, Si omis = l'URL est traduit

Description

La commande OUVRIR URL WEB lance votre browser Web et l'ouvre sur l'URL passé dans le paramètre url.

Si votre browser Web était déjà ouvert au moment de l'exécution de la commande :
• Sous Windows, une instance supplémentaire du browser est exécutée et affiche la page spécifiée par url.
• Sous MacOS, la page spécifiée par url remplace la page courante dans le browser.
S'il n'y a pas de browser sur les volumes connectés à l'ordinateur, la commande ne fait rien.

Note : Sous MacOS, cette commande tient compte des réglages Internet (navigateur par défaut) définis dans le Gestionnaire de configuration ou Configuration Manager, lorsque ceux-ci existent.

4D encode automatiquement les caractères spéciaux de l'URL. Si vous passez le caractère *, 4D ne traduira pas les caractères spéciaux de l'URL. Cette option permet d'accéder à ou de renvoyer des URL du type "http://www.server.net/page.htm?q=quelquechose".

Note : Cette commande ne fonctionne pas dans le cadre d'un process Web.

Exemples

(1) Une fois l'instruction suivante exécutée, le browser Web est lancé et l'URL est traduit par "file:///D%3A/web%20fichier.htm"

⇒ **OUVRIR URL WEB**("file:///D:/web fichier.htm")

(2) Une fois l'instruction suivante exécutée, le browser Web est lancé et l'URL reste "file:///D:/web fichier.htm"

⇒ **OUVRIR URL WEB**("file:///D:/web fichier.htm";*)

(3) L'instruction suivante lance le browser et le connecte à la page d'accueil du site Web de 4D :

⇒ **OUVRIR URL WEB**("http://www.4D.fr/")

50

Sous-enregistrements

CREER SOUS ENREGISTREMENT (sousTable)

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table dans laquelle vous voulez créer un sous-enregistrement

Description

CREER SOUS ENREGISTREMENT crée un nouveau sous-enregistrement dans sousTable et en fait le sous-enregistrement courant. Ce nouveau sous-enregistrement n'est sauvegardé que lorsque l'enregistrement parent est lui-même sauvegardé. L'enregistrement parent peut être sauvegardé par une commande telle que STOCKER ENREGISTREMENT ou lorsque l'utilisateur le valide. S'il n'y a pas d'enregistrement courant, CREER SOUS ENREGISTREMENT ne fait rien. Pour ajouter un nouveau sous-enregistrement dans un formulaire de saisie de sous-enregistrements, utilisez AJOUTER SOUS ENREGISTREMENT.

Exemple

L'exemple suivant est la méthode objet d'un bouton. Lorsqu'elle est exécutée (lorsque l'utilisateur clique sur le bouton), elle crée de nouveaux sous-enregistrements pour des enfants. La boucle Repeter permet à l'utilisateur d'ajouter plusieurs enfants, jusqu'à ce qu'il clique sur Annuler. Le formulaire fait apparaître les enfants dans un sous-formulaire, mais ne permet pas d'y saisir directement des données car l'option "Saisissable" a été désactivée :

```
Repeter
      ` Répéter jusqu'à ce que l'utilisateur clique sur Annuler
   vEnfant := Demander("Prénom (annuler si terminé) :")
   Si (OK = 1)
         ` Création d'un nouveau sous-enregistrement pour un enfant
⇒     CREER SOUS ENREGISTREMENT([Personnes]Enfants)
         ` Assignation du prénom de l'enfant au sous-champ
      [Personnes]Enfants'Prénom := vEnfant
   Fin de si
Jusque (OK = 0)
```

Référence

AJOUTER SOUS ENREGISTREMENT, STOCKER ENREGISTREMENT, SUPPRIMER SOUS ENREGISTREMENT.

SUPPRIMER SOUS ENREGISTREMENT (sousTable)

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table de laquelle supprimer le sous-enregistrement courant

Description

SUPPRIMER SOUS ENREGISTREMENT supprime le sous-enregistrement courant de sousTable. S'il n'y a pas de sous-enregistrement courant, SUPPRIMER SOUS ENREGISTREMENT ne fait rien. Après la suppression, la sous-sélection courante de sousTable est vide. En conséquence, la commande SUPPRIMER SOUS ENREGISTREMENT ne peut pas être utilisée pour parcourir une sous-sélection et supprimer des sous-enregistrements sélectionnés.

La suppression d'un sous-enregistrement n'est pas définitive tant que l'enregistrement parent n'est pas validé. La suppression d'un enregistrement parent entraîne automatiquement la suppression de tous ses sous-enregistrements.

Pour supprimer une sous-sélection, vous devez d'abord créer la sous-sélection, puis supprimer le premier sous-enregistrement, recréer la sous-sélection puis supprimer de nouveau le premier enregistrement, etc.

Exemples

(1) L'exemple suivant supprime tous les sous-enregistrements d'une sous-table :

```
   TOUS LES SOUS ENREGISTREMENTS([Personnes]Enfants)
   Tant que (Sous enregistrements trouves([Personnes]Enfants)>0)
⇒     SUPPRIMER SOUS ENREGISTREMENT([Personnes]Enfants)
      TOUS LES SOUS ENREGISTREMENTS([Personnes]Enfants)
   Fin tant que
```

(2) L'exemple suivant supprime de la sous-table [Personnes]Enfants tous les sous-enregistrements dans lesquels l'âge des enfants est supérieur ou égal à 12 ans :

```
TOUT SELECTIONNER([Personnes]) ` Sélection de tous les enregistrements
   ` Pour chaque enregistrement
Boucle ($vELEnrg;1;Enregistrements trouves([Personnes]))
      ` Recherche de tous les enregistrements ayant des sous-enregistrements
      ` correspondant aux critères
   CHERCHER SOUS ENREGISTREMENTS([Personnes]Enfants;
                                 [Personnes]Enfants'Age>=12)
      ` Boucle jusqu'à ce que la recherche ne trouve plus d'enregistrement
   Tant que (Sous enregistrements trouves([Personnes]Enfants)>0)
         ` Supprimer le sous-enregistrement
⇒     SUPPRIMER SOUS ENREGISTREMENT([Personnes]Enfants)
         ` On poursuit la recherche
      CHERCHER SOUS ENREGISTREMENTS([Personnes]Enfants;
                                    [Personnes]Enfants'Age>=12)
   Fin tant que
   STOCKER ENREGISTREMENT([Personnes]) ` Sauvegarde de l'enregistrement parent
   ENREGISTREMENT SUIVANT([Personnes])
Fin de boucle
```

Référence

CHERCHER SOUS ENREGISTREMENTS, Sous enregistrements trouves, STOCKER ENREGISTREMENT, TOUS LES SOUS ENREGISTREMENTS.

TOUS LES SOUS ENREGISTREMENTS (sousTable)

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table dans laquelle sélectionner tous les sous-enregistrements

Description

TOUS LES SOUS ENREGISTREMENTS fait de tous les sous-enregistrements de sousTable la sous-sélection courante. S'il n'existe aucun enregistrement parent courant, TOUS LES SOUS ENREGISTREMENTS ne fait rien. Lorsqu'un enregistrement parent est chargé depuis le disque, la sous-sélection courante contient tous les sous-enregistrements. Une sous-sélection peut ne plus contenir tous les sous-enregistrements après un appel aux commandes AJOUTER SOUS ENREGISTREMENT, CHERCHER SOUS ENREGISTREMENTS, ou SUPPRIMER SOUS ENREGISTREMENT.

Exemple

Dans l'exemple suivant, on sélectionne tous les sous-enregistrements afin d'être sûr qu'ils soient tous inclus dans le total :

```
⇒    TOUS LES SOUS ENREGISTREMENTS ([Stats]Ventes)
     VentesTotales := Somme ([Stats]Ventes'Francs)
```

Référence

CHERCHER SOUS ENREGISTREMENTS, Sous enregistrements trouves.

Sous enregistrements trouves (sousTable) → Numérique

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table dont vous voulez connaître le nombre de sous-enregistrements
Résultat	Numérique	←	Nombre de sous-enregistrements de la sous-sélection courante

Description

Sous enregistrements trouves retourne le nombre de sous-enregistrements de la sous-sélection courante de sousTable. Cette fonction s'applique uniquement aux sous-enregistrements de l'enregistrement courant. Elle est en quelque sorte l'équivalent de la fonction Enregistrements trouves pour les sous-enregistrements. Le résultat retourné est indéfini s'il n'existe pas d'enregistrement parent.

Exemple

L'exemple suivant sélectionne tous les sous-enregistrements puis affiche le nombre d'enfants pour l'enregistrement parent :

```
      ` Sélection de tous les enfants, puis affichage de leur nombre
      TOUS LES SOUS ENREGISTREMENTS ([Personnes]Enfants)
⇒     ALERTE ("Le nombre d'enfants est : "+Chaine(Sous enregistrements trouves
                                                  ([Personnes]Enfants)))
```

Référence

CHERCHER SOUS ENREGISTREMENTS, TOUS LES SOUS ENREGISTREMENTS.

APPLIQUER A SOUS SELECTION (sousTable; formule)

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table à laquelle appliquer la formule
formule	Formule	→	Ligne de code ou méthode

Description

APPLIQUER A SOUS SELECTION applique formule à chaque sous-enregistrement de la sous-sélection courante de sousTable. formule peut être une ligne d'instructions ou une méthode. Si formule entraîne la modification d'un sous-enregistrement, le sous-enregistrement modifié n'est sauvegardé que lorsque l'enregistrement parent est sauvegardé. Si la sous-sélection courante est vide, APPLIQUER A SOUS SELECTION ne fait rien.

APPLIQUER A SELECTION peut être utilisé pour récupérer et traiter des informations d'une sous-sélection d'enregistrements ou pour modifier une sous-sélection.

Exemple

L'exemple suivant met une lettre capitale aux prénoms du champ [Personnes]Enfants.

```
     TOUT SELECTIONNER ([Personnes]Enfants)
⇒    APPLIQUER A SOUS SELECTION([Personnes]Enfants;[Personnes]Enfants'Nom:=
                                   Majusc(Sous chaine([Personnes]Enfants'Nom;1;1))+
                                     Minusc(Sous chaine([Personnes]Enfants'Nom;2)))
```

Note : La formule doit être écrite sur une seule ligne dans l'éditeur de méthodes de 4D.

Référence

CHERCHER SOUS ENREGISTREMENTS, STOCKER ENREGISTREMENT, TOUS LES SOUS ENREGISTREMENTS.

DEBUT SOUS ENREGISTREMENT (sousTable)

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table de laquelle charger le premier sous-enregistrement de la sélection courante

Description

DEBUT SOUS ENREGISTREMENT charge le premier sous-enregistrement de la sélection courante de sousTable et en fait le sous-enregistrement courant. Toutes les commandes de recherche, de sélection et de tri font également du premier sous-enregistrement le sous-enregistrement courant. Si la sous-sélection courante est vide, DEBUT SOUS ENREGISTREMENT ne fait rien.

Exemple

L'exemple suivant met bout à bout les prénoms et les noms des enfants stockés dans une sous-table, puis les copie dans un tableau, nommé ttNoms :

```
      ` Création d'un tableau pour recevoir les noms
      TABLEAU TEXTE (ttNoms; Sous enregistrements trouves ([Personnes]Enfants))
         ` Commençons au premier sous-enregistrement
⇒     DEBUT SOUS ENREGISTREMENT ([Personnes]Enfants)
      Boucle ($vlS; 1; Sous enregistrements trouves ([Personnes]Enfants))
            ` Effectuons une boucle par enfant
         ttNoms{$vlS} := [Personnes]Enfants'Prénom+ " " + [Personnes]Enfants'Nom
         SOUS ENREGISTREMENT SUIVANT ([Personnes]Enfants)
      Fin de boucle
```

Référence

ALLER A DERNIER SOUS ENREGISTREMENT, SOUS ENREGISTREMENT PRECEDENT, SOUS ENREGISTREMENT SUIVANT.

ALLER A DERNIER SOUS ENREGISTREMENT (sousTable)

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table dans laquelle aller au dernier sous-enregistrement

Description

ALLER A DERNIER SOUS ENREGISTREMENT sélectionne le dernier sous enregistrement de la sous-sélection courante de sousTable et en fait le sous-enregistrement courant. Si la sous-sélection courante est vide, ALLER A DERNIER SOUS ENREGISTREMENT ne fait rien.

Exemple

L'exemple suivant concatène le prénom et le nom de famille de chaque enfant dans la sous-table. Les noms sont copiés dans un tableau, taNoms. Cet exemple ressemble à celui de la commande DEBUT SOUS ENREGISTREMENT mais celui-ci va du dernier sous-enregistrement au premier :

```
      ` Créer un tableau pour contenir les noms
   TABLEAU TEXTE (taNoms; Sous enregistrements trouves ([Personnes]Enfants))
⇒  ALLER A DERNIER SOUS ENREGISTREMENT ([Personnes]Enfants)
      ` Commencer au dernier sous enregistrement et boucler pour chaque enfant
   Boucle ($EL; 1; Sous enregistrements trouves ([Personnes]Enfants))
      Noms{$EL} := [Personnes]Enfants'Prénom + " " + [Personnes]Enfants'Nom
      SOUS ENREGISTREMENT PRECEDENT ([Personnes]Enfants)
   Fin de boucle
```

Référence

DEBUT SOUS ENREGISTREMENT, SOUS ENREGISTREMENT PRECEDENT, SOUS ENREGISTREMENT SUIVANT.

SOUS ENREGISTREMENT SUIVANT (sousTable)

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table dans laquelle se placer sur le sous-enregistrement suivant de la sélection courante

Description

SOUS ENREGISTREMENT SUIVANT place le pointeur de sous-enregistrement courant sur le sous-enregistrement suivant de la sous-sélection courante de sousTable. Si SOUS ENREGISTREMENT SUIVANT place le pointeur de sous-enregistrement courant après le dernier sous-enregistrement, Fin sous enregistrement retourne VRAI, et il n'y a plus de sous-enregistrement courant.
Lorsque Fin sous enregistrement retourne VRAI, utilisez DEBUT SOUS ENREGISTREMENT ou ALLER A DERNIER SOUS ENREGISTREMENT pour replacer le pointeur de sous-enregistrement courant dans la sous-sélection courante. Si la sous-sélection courante est vide, ou si Avant sous enregistrement retourne VRAI, SOUS ENREGISTREMENT SUIVANT ne fait rien.

Exemple

Reportez-vous à l'exemple de la commande DEBUT SOUS ENREGISTREMENT.

Référence

ALLER A DERNIER SOUS ENREGISTREMENT, DEBUT SOUS ENREGISTREMENT, SOUS ENREGISTREMENT PRECEDENT.

SOUS ENREGISTREMENT PRECEDENT (sousTable)

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table dans laquelle se placer sur le sous-enregistrement précédent de la sélection courante

Description

SOUS ENREGISTREMENT PRECEDENT place le pointeur de sous-enregistrement courant sur le sous-enregistrement précédent dans la sous-sélection courante de sousTable. Si SOUS ENREGISTREMENT PRECEDENT place les pointeur avant le premier sous-enregistrement, Avant sous enregistrement retourne Vrai, et il n'y a plus de sous-enregistrement courant. Dans ce cas, utilisez DEBUT SOUS ENREGISTREMENT ou ALLER A DERNIER SOUS ENREGISTREMENT pour replacer le pointeur dans la sous-sélection courante.
Si la sous-sélection courante est vide ou si Fin sous enregistrement retourne Vrai, SOUS ENREGISTREMENT PRECEDENT ne fait rien.

Exemple

Reportez-vous à l'exemple de la commande ALLER A DERNIER SOUS ENREGISTREMENT.

Référence

ALLER A DERNIER SOUS ENREGISTREMENT, DEBUT SOUS ENREGISTREMENT, SOUS ENREGISTREMENT SUIVANT.

Avant sous enregistrement (sousTable) → Booléen

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table pour laquelle vous testez si le pointeur se trouve avant la sous-sélection
Résultat	Booléen	←	Avant la sous-sélection (Vrai), sinon (Faux)

Description

Avant sous enregistrement retourne Vrai lorsque le pointeur de sous-enregistrement courant se trouve avant le premier sous-enregistrement de sousTable. Avant sous enregistrement est utilisé pour vérifier si la commande SOUS ENREGISTREMENT PRECEDENT a déplacé le pointeur avant le premier sous-enregistrement. Si la sous-sélection courante est vide, Avant sous enregistrement retourne Vrai.

Exemple

L'exemple est la méthode objet d'un bouton. Lorsque l'utilisateur clique sur le bouton, le pointeur se place sur le sous-enregistrement précédent. Si le pointeur se trouve avant le premier sous-enregistrement, il est replacé sur le dernier sous-enregistrement :

```
    ` Aller au sous-enregistrement précédent
SOUS ENREGISTREMENT PRECEDENT ([Personnes]Enfants)
⇒   Si (Avant sous enregistrement ([Personnes]Enfants)  ` Si on y est déjà
       ALLER A DERNIER SOUS ENREGISTREMENT ([Personnes]Enfants)
          ` Aller au dernier sous-enregistrement
    Fin de si
```

Référence

SOUS ENREGISTREMENT PRECEDENT.

Fin sous enregistrement (sousTable) → Booléen

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table pour laquelle tester si le pointeur de sous-enregistrement courant est placé au-delà du dernier sous-enregistrement sélectionné
Résultat	Booléen	←	Oui (Vrai) ou Non (Faux)

Description

Fin sous enregistrement retourne VRAI lorsque le pointeur de sous-enregistrement courant se trouve après la fin de la sous-sélection courante de sousTable. Fin sous enregistrement a pour rôle de tester si l'utilisation de la commande SOUS ENREGISTREMENT SUIVANT place ou non le pointeur derrière le dernier sous-enregistrement de la sélection. Si la sous-sélection courante est vide, Fin sous enregistrement retourne Vrai.

Exemple

L'exemple suivant est la méthode objet d'un bouton. Lorsque l'utilisateur clique sur le bouton, le pointeur se place sur le sous-enregistrement suivant. Si le pointeur se retrouve derrière le dernier sous-enregistrement, il est replacé sur le premier sous-enregistrement :

```
    ` Aller au sous-enregistrement suivant
SOUS ENREGISTREMENT SUIVANT ([Personnes]Enfants)
Si (Fin sous enregistrement ([Personnes]Enfants)) ` Si nous sommes allés trop loin...
    ` Aller au premier sous-enregistrement
  DEBUT SOUS ENREGISTREMENT ([Personnes]Enfants)
Fin de si
```

Référence

SOUS ENREGISTREMENT SUIVANT.

TRIER SOUS ENREGISTREMENTS (sousTable; sousChamp{; direction}{; sousChamp2; direction2; ...; sousChampN; directionN})

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table contenant le(s) sous-champ(s) à trier
sousChamp	Sous-champ	→	Sous-champ sur lequel effectuer le tri
direction	> ou <	→	> tri croissant ou < tri décroissant

Description

TRIER SOUS ENREGISTREMENTS trie la sous-sélection courante de sousTable. Seule la sous-sélection de sousTable contenue dans l'enregistrement parent courant est triée.

Le paramètre direction définit le sens du tri de sousChamp : ascendant ou descendant. Si direction a pour valeur le symbole “supérieur à” (>), les sous-enregistrements sont triés dans l'ordre ascendant. Si direction a pour valeur le symbole “inférieur à” (<), les sous-enregistrements sont triés dans l'ordre descendant. direction peut être omis (dans le cadre d'un tri sur un seul niveau) ; le tri est alors ascendant.

Vous pouvez spécifier plusieurs niveaux de tris en incluant autant de sous-champs et de symboles de tris que vous voulez.
Une fois le tri terminé, le premier sous-enregistrement de la sous-sélection triée devient le sous-enregistrement courant. Le tri des sous-enregistrements est une opération dynamique. Les sous-enregistrements ne sont jamais sauvegardés dans l'ordre où ils se trouvent après un tri.
S'il n'y a pas d'enregistrement courant ou de sous-enregistrement, TRIER SOUS ENREGISTREMENTS ne fait rien.
Si un formulaire contient un sous-formulaire dont l'impression est "limitée par le cadre", la commande n'a besoin d'être appelée qu'une seule fois avant l'impression, pendant l'événement formulaire Sur chargement de la méthode formulaire parente.

Exemple

L'exemple suivant trie la sous-table [Stats]Ventes dans un ordre croissant, sur la base du sous-champ VentesDollars :

```
⇒ TRIER SOUS ENREGISTREMENTS ([Stats]Ventes; [Stats]VentesDollars; >)
```

Référence

CHERCHER SOUS ENREGISTREMENTS.

CHERCHER SOUS ENREGISTREMENTS (sousTable; formule)

Paramètre	Type		Description
sousTable	Sous-table	→	Sous-table dans laquelle effectuer la recherche
formule	Booléen	→	Formule de recherche

Description

CHERCHER SOUS ENREGISTREMENTS effectue une recherche dans sousTable et crée une nouvelle sous-sélection courante. CHERCHER SOUS ENREGISTREMENTS est la seule commande qui permet d'effectuer une recherche parmi des sous-enregistrements et qui retourne une sélection de sous-enregistrements. formule est appliquée à chaque sous-enregistrement de sousTable. Lorsque la formule est Vraie, le sous-enregistrement est ajouté à la nouvelle sous-sélection. Une fois que l'exécution de la formule est terminée, le premier sous-enregistrement de la sous-sélection devient le sous-enregistrement courant de sousTable.

N'oubliez pas que CHERCHER SOUS ENREGISTREMENTS effectue une recherche parmi les sous-enregistrements de la sous-table pour l'enregistrement parent courant uniquement, et non parmi tous les sous-enregistrements associés aux enregistrements de la table parente. CHERCHER SOUS ENREGISTREMENTS ne modifie pas l'enregistrement parent courant.

Typiquement, formule compare un sous-champ à une variable ou une constante, à l'aide d'un opérateur relationnel. formule peut comprendre plusieurs tests reliés par des opérateurs de type ET (&) ou de type OU (|). formule peut également être ou contenir une fonction. Le caractère Joker (@) peut être utilisé avec les arguments de type chaîne.

S'il n'y a pas d'enregistrement ni de sous-enregistrement courant, CHERCHER SOUS ENREGISTREMENTS ne fait rien.

Exemple

L'exemple suivant recherche les enfants âgés de plus de 10 ans :

```
⇒ CHERCHER SOUS ENREGISTREMENTS ([Personnes]Enfants;
                                 [Personnes]Enfants'Age>10)
```

Référence

Sous enregistrements trouves, TOUS LES SOUS ENREGISTREMENTS, TRIER SOUS ENREGISTREMENTS.

51

Table

TABLE PAR DEFAUT

Table

version 5

TABLE PAR DEFAUT (table)

Paramètre	Type		Description
table	Table	→	Table à définir comme table par défaut

Description

TABLE PAR DEFAUT définit table comme la table par défaut pour le process courant.

Un process n'a pas de table par défaut tant que la commande TABLE PAR DEFAUT n'a pas été exécutée. Après qu'une table par défaut ait été désignée, toute commande pour laquelle le paramètre table n'a pas été défini s'appliquera à la table par défaut. Considérez par exemple l'instruction suivante :

```
FORMULAIRE ENTREE ([Table];"Formulaire")
```

Si [Table] a préalablement été définie comme table par défaut, la même instruction pourrait s'écrire :

```
FORMULAIRE ENTREE ("Formulaire")
```

Une des raisons pour lesquelles vous pouvez définir une table par défaut est l'écriture de code qui ne soit pas lié à une table. Cela permet au même code d'être appliqué à différentes tables.
Vous pouvez aussi utiliser des pointeurs vers des tables pour écrire du code non lié aux tables. Pour plus d'informations sur cette technique, reportez-vous à la description de la commande Nom de la table.

TABLE PAR DEFAUT ne permet pas d'omettre les noms de tables lorsque vous vous référez à des champs. Par exemple :

```
[MaTable]MonChamp := "Une chaîne"    ` OK
```

ne peut pas s'écrire :

```
TABLE PAR DEFAUT ([MaTable])
MonChamp := "Une chaîne"     ` Incorrect
```

... simplement parce qu'une table par défaut a été définie. Toutefois, vous pouvez omettre le nom de la table lorsque vous vous référez à des champs dans des méthodes table, des formulaires et des objets appartenant à la table.

Dans 4e Dimension, toutes les tables sont “ouvertes” et prêtes à être utilisées. TABLE PAR DEFAUT n'ouvre pas de table, ne définit pas de table courante et ne prépare pas de table pour la saisie ou l'affichage. TABLE PAR DEFAUT est simplement une facilité de programmation proposée pour accélérer la saisie du code et le rendre plus facile à lire.

Conseil : Bien que l'appel de TABLE PAR DEFAUT et l'omission du nom de la table rendent le code plus lisible, la plupart des programmeurs estiment que l'utilisation de cette commande apporte plus d'inconvénients que d'avantages.

Exemple

L'exemple suivant présente la même méthode avec et sans la commande TABLE PAR DEFAUT. Le code est une boucle souvent utilisée pour créer de nouveaux enregistrements dans une base. Les commandes FORMULAIRE ENTREE et AJOUTER ENREGISTREMENT nécessitent le nom d'une table comme premier paramètre :

```
FORMULAIRE ENTREE ([Clients]; "Ajout Enrg")
Repeter
   AJOUTER ENREGISTREMENT ([Clients])
Jusque (OK = 0)
```

Voici le résultat lorsqu'une table par défaut est définie :

```
⇒  TABLE PAR DEFAUT ([Clients])
   FORMULAIRE ENTREE ("Ajout Enrg")
   Repeter
      AJOUTER ENREGISTREMENT
   Jusque (OK = 0)
```

Référence

Table par defaut courante.

Table par defaut courante

Table par defaut courante → Pointeur

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Pointeur	←	Pointeur vers la table par défaut

Description

Table par defaut courante retourne un pointeur vers la table qui a été passée au dernier appel de la commande TABLE PAR DEFAUT pour le process courant.

Exemple

La ligne de code suivante inscrit le nom de la table courante par défaut dans le titre de la fenêtre :

⇒ **CHANGER TITRE FENETRE**(**Nom de la table**(**Table par defaut courante**))

Référence

Nom de la table, Table, TABLE PAR DEFAUT.

FORMULAIRE ENTREE

Table

version 6.0 (Modifiée)

FORMULAIRE ENTREE ({table; }formulaire{; *})

Paramètre	Type		Description
table	Table	→	Table pour laquelle définir le formulaire entrée ou Table par défaut si ce paramètre est omis
formulaire	Alpha	→	Nom du formulaire à utiliser
*		→	Taille de fenêtre automatique

Description

FORMULAIRE ENTREE désigne formulaire comme formulaire entrée courant de table pour le process courant. formulaire doit appartenir à table.

La portée de cette commande est le process courant. Chaque table dispose d'un formulaire entrée courant pour chaque process.

La commande FORMULAIRE ENTREE n'affiche pas de formulaire ; elle désigne juste celui qui sera affiché ou utilisé par d'autres commandes. Pour plus d'informations sur la création de formulaires, reportez-vous au manuel *Mode Structure* de 4e Dimension.

Pour chaque table de la base, un formulaire entrée par défaut est défini dans la fenêtre de l'Explorateur en mode Structure. Ce formulaire est utilisé si la commande FORMULAIRE ENTREE n'est pas appelée, ou si le paramètre formulaire est invalide.

Le formulaire entrée est affiché par de nombreuses commandes. Ces commandes sont généralement utilisées pour la saisie ou la modification de valeurs. Les commandes suivantes affichent un formulaire entrée :

- AJOUTER ENREGISTREMENT
- CHERCHER PAR EXEMPLE
- AFFICHER ENREGISTREMENT
- MODIFIER ENREGISTREMENT

Les commandes VISUALISER SELECTION et MODIFIER SELECTION affichent une liste d'enregistrements dans le formulaire sortie. Chacune d'entre elles permet ensuite à l'utilisateur de double-cliquer sur un enregistrement, qui s'affiche alors dans le formulaire entrée.

Le formulaire entrée est aussi utilisé par les commandes d'import LECTURE DIF, LECTURE SYLK et LECTURE ASCII.

Le paramètre optionnel * est destiné à être utilisé conjointement avec les propriétés du formulaire, que vous définissez en mode Structure dans la fenêtre des Propriétés du formulaire, et avec la commande Creer fenetre. En passant le paramètre *, vous indiquez à 4D d'utiliser les propriétés du formulaire pour redimensionner automatiquement la fenêtre lors de son utilisation ultérieure comme formulaire entrée ou comme dialogue. Reportez-vous à la description de la commande Creer fenetre pour plus d'informations sur ce point.

Note : Que vous passiez ou non le paramètre *, FORMULAIRE ENTREE change le formulaire entrée pour la table.

Exemple

L'exemple suivant illustre une utilisation typique de FORMULAIRE ENTREE. A noter que, si dans cet exemple FORMULAIRE ENTREE est appelé juste avant que le formulaire soit utilisé, cela n'est absolument pas nécessaire. La commande FORMULAIRE ENTREE peut en fait être appelée dans une autre méthode, du moment qu'elle est exécutée avant celle-ci :

```
      ` Formulaire pour les nouvelles sociétés
⇒     FORMULAIRE ENTREE ([Sociétés]; "Nouvelle Sté")
      AJOUTER ENREGISTREMENT ([Sociétés]) ` Ajout d'une nouvelle société
```

Référence

AFFICHER ENREGISTREMENT, AJOUTER ENREGISTREMENT, CHERCHER PAR EXEMPLE, Creer fenetre, FORMULAIRE SORTIE, LECTURE ASCII, LECTURE DIF, LECTURE SYLK, MODIFIER ENREGISTREMENT, MODIFIER SELECTION, VISUALISER SELECTION.

FORMULAIRE SORTIE

Table

version 5

FORMULAIRE SORTIE ({table; }formulaire)

Paramètre	Type		Description
table	Table	→	Table pour laquelle définir le formulaire sortie ou Table par défaut si ce paramètre est omis
formulaire	Alpha	→	Nom du formulaire

Description

FORMULAIRE SORTIE vous permet de définir formulaire comme formulaire sortie courant de table pour le process courant. formulaire doit appartenir à table.

La portée de cette commande est le process courant. Chaque table dispose de son propre formulaire sortie dans chaque process.

La commande FORMULAIRE SORTIE ne provoque pas l'affichage du formulaire ; elle désigne simplement le formulaire devant être imprimé, affiché, ou utilisé par d'autres commandes. Pour plus d'informations sur la création de formulaires, reportez-vous au manuel *Mode Structure* de 4e Dimension.

Le formulaire sortie par défaut est défini dans la fenêtre de l'Explorateur du mode Structure pour chaque table. Il est identifié par la lettre S placée près de son nom dans l'Explorateur et dans les boîtes de dialogue listant les formulaires. Ce formulaire par défaut sera utilisé si vous n'appelez pas la commande FORMULAIRE SORTIE ou si vous passez à cette commande un nom de formulaire erroné ou inexistant.

Les formulaires sortie sont exploités par trois groupes de commandes. Le premier groupe gère l'affichage des enregistrements à l'écran, le deuxième gère la génération d'états et le troisième gère l'export de données.

Chacune des commandes suivantes affiche une liste d'enregistrements dans un formulaire sortie :
- VISUALISER SELECTION
- MODIFIER SELECTION

Vous utilisez le formulaire sortie lorsque vous créez des états à l'aide des commandes suivantes :
• IMPRIMER ETIQUETTES
• IMPRIMER SELECTION

Chacune des commandes d'export suivantes utilise également le formulaire sortie :
• ECRITURE DIF
• ECRITURE SYLK
• ECRITURE ASCII

Exemple

L'exemple suivant illustre une utilisation typique FORMULAIRE SORTIE. Notez que, bien que dans cet exemple la commande FORMULAIRE SORTIE soit placée juste avant que le formulaire soit utilisé, cela n'est pas obligatoire. En fait, la commande pourrait se trouver dans n'importe quelle autre méthode, dans la mesure où elle est exécutée avant celle-ci :

```
      FORMULAIRE ENTREE ([Parties]; "Saisie Parties") ` Sélection du formulaire entrée
⇒     FORMULAIRE SORTIE ([Parties]; "Liste Parties") ` Sélection du formulaire sortie
      MODIFIER SELECTION ([Parties]) ` Cette commande utilise les deux formulaires
```

Référence

ECRITURE ASCII, ECRITURE DIF, ECRITURE SYLK, FORMULAIRE ENTREE, IMPRIMER ETIQUETTES, IMPRIMER SELECTION, MODIFIER SELECTION, VISUALISER SELECTION.

Table du formulaire courant

Table

version 6.0

Table du formulaire courant → Pointeur

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Pointeur	←	Pointeur vers la table à laquelle appartient le formulaire actuellement affiché

Description

La fonction Table du formulaire courant retourne un pointeur vers la table à laquelle appartient le formulaire affiché à l'écran ou imprimé dans le process courant.

S'il n'y a pas de formulaire affiché ou en cours d'impression dans le process courant, la fonction retourne Nil.

Si plusieurs fenêtres sont ouvertes simultanément dans le process courant (ce qui signifie que la dernière fenêtre ouverte est la fenêtre courante active), la fonction retourne un pointeur vers la table du formulaire affiché dans la fenêtre active.

Si le formulaire affiché est le formulaire détaillé d'une zone de sous-formulaire (c'est-à-dire que pendant la saisie de données, l'utilisateur a double-cliqué sur un enregistrement ou un sous-enregistrement dans une zone de sous-formulaire "double-cliquable"), la fonction retourne :
• un pointeur vers la table de la zone de sous-formulaire, si cette dernière affiche une table.
• un pointeur non significatif si la zone de sous-formulaire affiche une sous-table.

Exemple

Dans votre application, vous utilisez la convention suivante : au moment de l'affichage d'un enregistrement, la variable vsEnrCourant, si elle est présente dans un formulaire, affiche "Nouvel enregistrement" si vous créez un nouvel enregistrement, ou par exemple "56 parmi 5200" si vous ouvrez le 56e enregistrement d'une sélection en comportant 5200. Pour cela, vous pouvez créer une fois la variable vsEnrCourant et lui associer la méthode objet décrite ci-dessous, puis la copier et la coller dans tous les formulaires que vous voulez :

```
      ` Méthode objet de la variable non saisissable vsEnrCourant
   Au cas ou
      : (Evenement formulaire =Sur chargement)
         C_ALPHA (31;vsEnrCourant)
         C_POINTEUR ($vpTableParente)
         C_ENTIER LONG ($vlNumEnr)
⇒        $vpTableParente:=Table du formulaire courant
         $vlNumEnr:=Numero enregistrement ($vpTableParente->)
         Au cas ou
            : ($vlNumEnr=-3)
               vsEnrCourant:="Nouvel enregistrement"
            : ($vlNumEnr=-1)
               vsEnrCourant:="Pas d'enregistrement"
            : ($vlNumEnr>=0)
               vsEnrCourant:=Chaine (Numero dans selection ($vpTableParente->))+
                        " parmi "+Chaine (Numero dans selection ($vpTableParente->))
         Fin de cas
   Fin de cas
```

Référence

DIALOGUE, FORMULAIRE ENTREE, FORMULAIRE SORTIE, IMPRIMER SELECTION.

52

Tableaux

Un **tableau** est une série ordonnée de variables de même type. Chaque variable est appelée un **élément** du tableau. La **taille** d'un tableau est le nombre d'éléments qu'il contient. La taille du tableau doit être définie au moment de sa création ; vous pouvez ensuite la modifier aussi souvent que nécessaire en ajoutant, insérant, ou supprimant des éléments, ou en appelant de nouveau la commande que vous avez utilisée pour créer le tableau.

Vous créez un tableau au moyen de l'une des commandes de déclaration de tableau. Pour plus d'informations, reportez-vous à la section Créer des tableaux.

Les éléments sont numérotés de **1 à N**, où N est la taille du tableau. Un tableau a toujours un **élément zéro**, auquel vous pouvez accéder tout comme vous accédez à n'importe quel autre élément du tableau, mais cet élément n'est pas affiché lorsqu'un tableau est placé dans un formulaire. Rien ne vous empêche cependant de l'utiliser avec le langage. Pour plus d'informations sur l'élément zéro, reportez-vous à la section Utiliser l'élément zéro d'un tableau.

Les tableaux sont des variables 4D. Comme toute variable, un tableau a une portée définie et suit les règles du langage 4D, bien qu'il existe quelques différences spécifiques. Pour plus d'informations, reportez-vous aux sections Les tableaux et le langage 4D et Tableaux et Pointeurs.

Les tableaux sont des objets du langage : vous pouvez créer et utiliser des tableaux qui n'apparaîtront jamais à l'écran. Mais les tableaux sont également des objets d'interface utilisateur. Pour plus d'informations sur l'interaction entre les tableaux et les objets de formulaire, reportez-vous aux sections Tableaux et objets de formulaire et Zones de défilement groupées.

Les tableaux doivent être utilisés pour manipuler une certaine quantité de données pendant une période brève. En contrepartie, comme les tableaux résident en mémoire, ils sont d'une utilisation rapide et facile. Pour plus d'informations, reportez-vous à la section Tableaux et mémoire.

Vous créez un tableau au moyen de l'une des commandes de déclaration de tableau décrites dans ce chapitre. Voici la liste des commandes de déclaration de tableau :

Commande	**Crée ou redimensionne un tableau de** :
TABLEAU ENTIER	Entiers (sur 2 octets)
TABLEAU ENTIER LONG	Entiers (sur 4 octets)*
TABLEAU REEL	Réels
TABLEAU TEXTE	Textes (de 0 à 32.000 caractères par élément)**
TABLEAU ALPHA	Alphanumériques (de 0 à 255 caractères par élément)**
TABLEAU DATE	Dates
TABLEAU BOOLEEN	Booléens
TABLEAU IMAGE	Images
TABLEAU POINTEUR	Pointeurs

Chaque commande de déclaration de tableau peut créer ou redimensionner des tableaux à une ou à deux dimensions. Pour plus d'informations sur les tableaux à deux dimensions, reportez-vous à la section Tableaux à deux dimensions.

(*) Les tableaux de type Entier long vous permettent de manipuler les données de type Heure. Pour afficher dans un formulaire un tableau sous forme d'heures, appliquez à l'objet affichant le tableau le format d'affichage &/x, où x représente le numéro d'ordre du format souhaité dans la liste de formatage des heures. Par exemple, &/4 correspond au format Heure:Minute.

(**) La différence entre les tableaux de type Texte et les tableaux de type Alphanumérique est la nature de leurs éléments. Dans ces deux types de tableaux, les éléments peuvent prendre des valeurs de texte (des caractères). Cependant :
• Dans un tableau Texte, chaque élément est de longueur variable et stocke ses caractères dans des parties différentes de la mémoire.
• Dans un tableau Alpha, tous les éléments ont la même longueur fixe (la longueur définie lors de la création du tableau). Tous les éléments sont stockés les uns à la suite des autres dans la même partie de la mémoire.

Cette différence structurelle rend les tableaux Alpha plus rapides que les tableaux Texte. Rappelez-vous cependant qu'un élément de tableau Alpha ne peut dépasser 255 caractères.

Cette ligne de code crée (déclare) un tableau d'entiers de 10 éléments :

```
TABLEAU ENTIER(aiUnTableau;10)
```

Ensuite, cette ligne de code redimensionne le même tableau à 20 éléments :

```
TABLEAU ENTIER(aiUnTableau;20)
```

Enfin, cette ligne de code redimensionne le même tableau à 0 élément :

```
TABLEAU ENTIER(aiUnTableau;0)
```

Vous référencez les éléments d'un tableau en utilisant des accolades ({...}). Un nombre entre accolades donne accès à l'adresse d'un élément particulier. Ce nombre est appelé **numéro de l'élément**. L'exemple ci-dessous place cinq noms dans le tableau nommé atNoms et les affiche ensuite dans une fenêtre d'alerte :

```
TABLEAU TEXTE (atNoms;5)
atNoms{1}  := "Richard"
atNoms{2}  := "Sarah"
atNoms{3}  := "Pierre"
atNoms{4}  := "Martine"
atNoms{5}  := "Jean"
Boucle ($vlElem;1;5)
   ALERTE ("L'élément #"+Chaine($vlElem)+" est égal à: "+atNoms{$vlElem})
Fin de boucle
```

Notez la syntaxe atNoms{$vlElem}. Au lieu de spécifier un nombre littéral comme atNoms{3}, vous pouvez utiliser une variable numérique indiquant à quel élément d'un tableau vous accédez.

Si vous utilisez les itérations permises par les structures répétitives (Boucle...Fin de boucle, Repeter...Jusque ou Tant que...Fin tant que), vous pouvez accéder à tout ou partie des éléments d'un tableau avec très peu de code.

Les tableaux et les autres commandes du langage 4D

Il existe d'autres commandes 4D qui permettent de créer ou de manipuler des tableaux. Pour plus d'informations, reportez-vous à la description des commandes suivantes :

• Pour travailler avec des tableaux et des sélections d'enregistrements, utilisez les commandes SELECTION LIMITEE VERS TABLEAU, SELECTION VERS TABLEAU, TABLEAU VERS SELECTION et VALEURS DISTINCTES.

• Vous pouvez créer des graphes à partir de valeurs stockées dans des tables, sous-tables, et des tableaux. Pour plus d'informations, reportez-vous à la commande GRAPHE.

• Bien que la version 6 apporte un jeu complet de commandes fonctionnant avec les listes hiérarchiques, les commandes ENUMERATION VERS TABLEAU et TABLEAU VERS ENUMERATION (de la version précédente de 4D) ont été conservées pour des raisons de compatibilité.

• De nouvelles commandes dans la version 6 construisent des tableaux en un appel. Ces commandes sont LISTE DES POLICES, LISTE FENETRES, LISTE DES VOLUMES, LISTE DES DOSSIERS, et LISTE DES DOCUMENTS.

Référence

Présentation des tableaux, TABLEAU ALPHA, TABLEAU BOOLEEN, TABLEAU DATE, TABLEAU ENTIER, TABLEAU ENTIER LONG, TABLEAU IMAGE, TABLEAU POINTEUR, TABLEAU REEL, TABLEAU TEXTE, Tableaux à deux dimensions.

Tableaux et objets de formulaire

Les tableaux sont des objets de langage — vous pouvez créer et utiliser des tableaux qui n'apparaîtront jamais à l'écran. Cependant, les tableaux sont aussi des objets d'interface utilisateur. Voici les types d'**objets de formulaire** gérés par des tableaux :

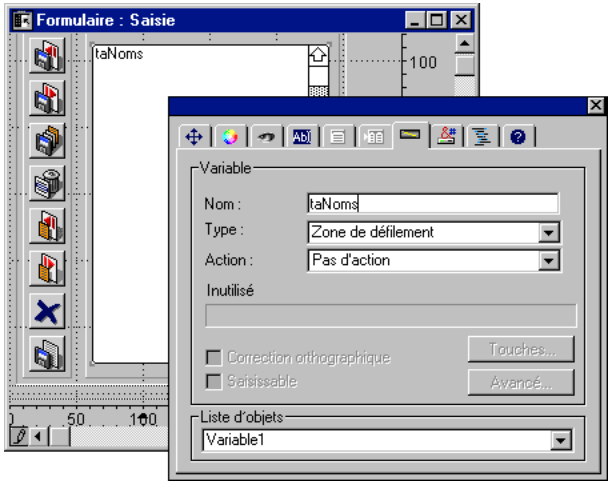
- Pop-up/Liste déroulante
- Combo Box
- Zone de défilement
- Onglet

Si vous pouvez prédéfinir ces objets dans l'éditeur de formulaires (en utilisant le bouton des valeurs par défaut de la fenêtre de propriétés des objets), vous pouvez également les définir par programmation, en utilisant les commandes de tableaux. Dans les deux cas, l'objet formulaire est géré par un tableau, créé par vous ou par 4D.

En utilisant ces objets, vous pouvez détecter quel élément de l'objet a été sélectionné (ou a reçu un clic souris) en testant l'**élément sélectionné** du tableau. Inversement, vous pouvez sélectionner un élément de l'objet en désignant l'élément de tableau correspondant.

Quand un tableau est utilisé pour gérer un objet de formulaire, il a une nature double ; il est à la fois un objet de langage et un objet d'interface utilisateur.

Par exemple, créez un formulaire, dans lequel vous placez une zone de défilement ; dans la page **Variable** de la fenêtre des propriétés de l'objet, vous nommez la variable objet :



Le nom, ici taNoms, est le nom du tableau que vous utilisez pour créer et gérer le tableau.

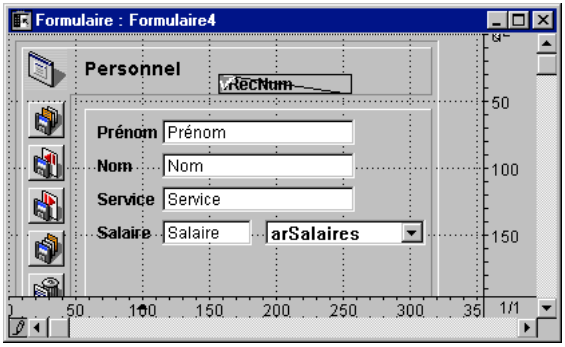
Note : Vous ne pouvez pas afficher des tableaux à deux dimensions ni des tableaux de pointeurs.

Exemple : création d'une liste déroulante

L'exemple suivant montre comment remplir un tableau et l'afficher dans une liste déroulante. Un tableau arSalaires est créé au moyen de la commande TABLEAU REEL. Il contient tous les salaires des personnes dans une entreprise américaine. Lorsque l'utilisateur sélectionne un élément dans la liste déroulante, le champ [Personnel]Salaire reçoit la valeur choisie dans le mode Utilisation ou Menus créés.

Création de la liste déroulante arSalaires dans un formulaire

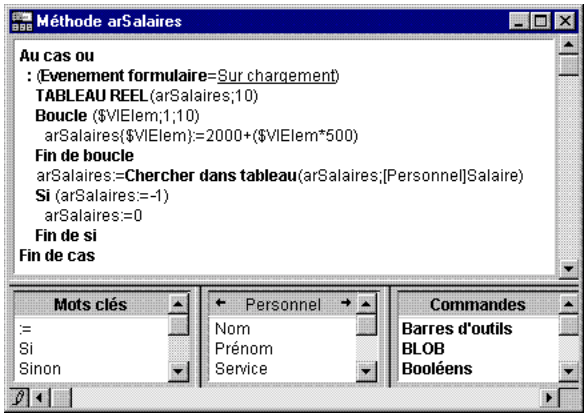
Créez une liste déroulante et nommez-là arSalaires. Le nom de la liste déroulante doit être le même que celui du tableau.



Initialisation du tableau

Initialisez le tableau arSalaires en spécifiant l'événement Sur chargement pour l'objet. Pour cela, n'oubliez pas d'activer cet événement dans la fenêtre des propriétés de l'objet, comme illustré ci-dessous :

Cliquez sur le bouton **Méthode objet...** et écrivez la méthode suivante :



Les lignes :

```
TABLEAU REEL(arSalaires;10)
Boucle($vlElem;1;10)
    arSalaires{$vlElem} :=2000+($vlElem*500)
Fin de boucle
```

... créent le tableau numérique 2500, 3000... 7000, correspondant aux salaires annuels allant de $30 000 à $84 000, avant impôts.

Les lignes :

```
arSalaires:=Chercher dans tableau(arSalaires;[Personnel]Salaire)
Si (arSalaires=-1)
   arSalaires:=0
Fin de si
```

... gèrent à la fois la création d'un nouvel enregistrement et la modification d'un enregistrement existant.

• Si vous créez un nouvel enregistrement, le champ [Personnel]Salaire est initialement égal à zéro. Dans ce cas, Chercher dans tableau ne trouve pas la valeur dans le tableau et retourne -1. Le test Si (arSalaires=-1) remet arSalaires à zéro, indiquant qu'aucun élément n'est sélectionné dans la liste déroulante.
• Si vous modifiez un enregistrement existant, Chercher dans tableau récupère la valeur dans le tableau et affecte à l'élément sélectionné de la liste déroulante la valeur courante du champ. Si la valeur pour un employé n'est pas dans la liste, le test Si (arSalaires=-1) désélectionne tous les éléments de la liste.

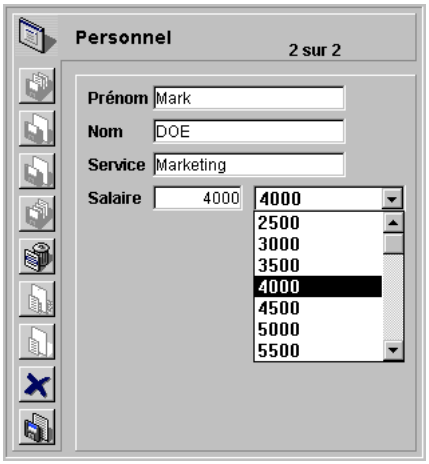
Note : Pour plus d'informations concernant l'**élément de tableau sélectionné**, lisez les paragraphes suivants.

Assignation de la valeur sélectionnée au champ [Personnel]Salaire
Pour reporter la valeur sélectionnée dans la liste déroulante arSalaires, il vous suffit de gérer l'événement Sur clic souris de l'objet. Le numéro de l'élément sélectionné est la valeur du tableau arSalaires. En conséquence, l'expression arSalaires{arSalaires} retourne la valeur choisie dans la liste déroulante.

Complétez ainsi la méthode de l'objet arSalaires :

```
Au cas ou
   : (Evenement formulaire=Sur chargement)
      TABLEAU REEL(arSalaires;10)
      Boucle($vlElem;1;10)
         arSalaires{$vlElem} :=2000+($vlElem*500)
      Fin de boucle
      arSalaires:=Chercher dans tableau(arSalaires;[Personnel]Salaire)
      Si (arSalaires=-1)
         arSalaires:=0
      Fin de si
   : (Evenement formulaire=Sur clic souris)
      [Personnel]Salaire:=arSalaires{arSalaires}
Fin de cas
```

En mode Utilisation ou Menus créés, la liste déroulante se présente ainsi :



Les paragraphes suivants décrivent les opérations élémentaires que vous pouvez effectuer sur les tableaux lorsque vous les utilisez comme objets de formulaire.

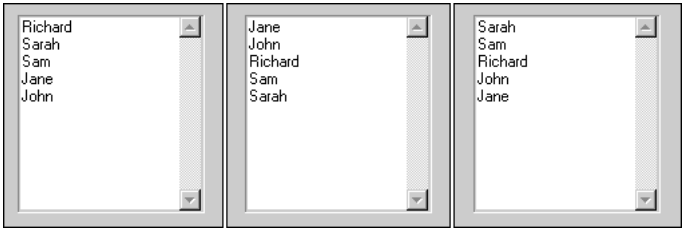
Obtenir la taille d'un tableau

Vous pouvez obtenir la taille courante d'un tableau au moyen de la commande Taille tableau. Si on reprend l'exemple précédent, la ligne de code qui suit affichera 5:

ALERTE ("La taille du tableau taNoms est: "+**Chaine**(**Taille tableau**(taNoms)))

Trier (réordonner) les éléments du tableau

Vous pouvez réordonner les éléments d'un tableau au moyen de la commande TRIER TABLEAU. Si on reprend l'exemple précédent, et étant entendu que le tableau est affiché comme une liste déroulante, vous pourrez voir ceci :



(a) Sur la gauche, la liste telle qu'elle se présente initialement.

(b) Au centre, la liste telle qu'elle se présente après l'exécution de la ligne de code suivante :

```
TRIER TABLEAU(taNoms;>)
```

(c) Sur la droite, la liste telle qu'elle se présente après l'exécution de la ligne de code suivante :

```
TRIER TABLEAU(taNoms;<)
```

Ajouter et supprimer des éléments

Vous pouvez ajouter, insérer, ou supprimer des éléments de tableau au moyen des commandes INSERER LIGNES et SUPPRIMER LIGNES.

Gestion des clics dans un tableau : test de l'élément sélectionné

Si on reprend l'exemple précédent, étant entendu que le tableau est affiché en tant que liste déroulante, vous pouvez gérer les clics souris de la manière suivante :

```
   ` Méthode objet liste déroulante taNoms
Au cas ou
   : (Evenement formulaire=Sur chargement)
         ` Initialiser le tableau (comme vu précédemment)
      TABLEAU TEXTE (taNoms;5)
         ` ...
   : (Evenement formulaire=Sur libération)
         ` Nous n'avons plus besoin du tableau
      EFFACER VARIABLE(taNoms)

   : (Evenement formulaire=Sur clic souris)
      Si (taNoms#0)
         vtInfo:="Vous avez cliqué sur : "+taNoms{taNoms}
      Fin de si
   : (Evenement formulaire=Sur double clic souris)
      Si (taNoms#0)
         ALERTE ("Vous avez double-cliqué sur : "+taNoms{taNoms}
      Fin de si
Fin de cas
```

Note : Les événements doivent avoir été activés dans la fenêtre des Propriétés de l'objet.

Alors que la syntaxe taNoms{$vlElem} vous permet de travailler sur un élément particulier du tableau, la syntaxe taNoms retourne le numéro de l'élément sélectionné dans le tableau. Ainsi, la syntaxe taNoms{taNoms} signifie “la valeur de l'élément sélectionné dans le tableau taNoms.” Si aucun élément n'est sélectionné, taNoms est égal à 0 (zéro). Le test Si (taNoms#0) détecte si un élément est effectivement sélectionné ou non.

Désigner l'élément sélectionné

Vous pouvez changer par programmation l'élément sélectionné en assignant une valeur au tableau. Exemples:

```
   ` Sélectionner le premier élément (si le tableau n'est pas vide)
taNoms:=1

   ` Sélectionner le dernier élément (si le tableau n'est pas vide)
taNoms:=Taille tableau(taNoms)

   ` Déselectionner l'élément sélectionné (s'il y en a un), plus aucun élément n'est
   ` alors sélectionné
taNoms:=0

Si ((0<taNoms)&(taNoms<Taille tableau(taNoms))
      ` Si possible, sélectionner l'élément suivant l'élément sélectionné
   taNoms:=taNoms+1
Fin de si

Si (1<taNoms)
      ` Si possible, sélectionner l'élément précédent l'élément sélectionné
   taNoms:=taNoms-1
Fin de si
```

Recherche d'une valeur dans le tableau

La commande Chercher dans tableau recherche une valeur particulière dans un tableau. Si nous reprenons l'exemple précédent, voici le code qui sélectionnera l'élément dont la valeur est “Richard”, si c'est ce que vous saisissez dans la boîte de dialogue de demande :

```
$vsNom:=Demander("Saisissez un prénom :")
Si (OK=1)
   $vlElem:=Chercher dans tableau (taNoms;$vsNom)
   Si ($vlElem>0)
      taNoms:=$vlElem
   Sinon
      ALERTE ("Il n'y a pas de "+$vsName+" dans cette liste de prénoms.")
   Fin de si
Fin de si
```

Les pop-up menus, listes déroulantes, zones de défilement et les onglets peuvent généralement être gérés de la même manière. Bien entendu, aucun code supplémentaire n'est nécessaire pour le redessinement des objets à l'écran à chaque fois que vous modifiez la valeur d'un élément, en ajoutez ou en supprimez.

Note : Pour créer et utiliser des onglets avec des icônes ainsi que des onglets activés ou désactivés, vous devez utiliser une liste hiérarchique comme objet de gestion associé à l'onglet. Pour plus d'informations, reportez-vous à l'exemple de la commande Nouvelle liste.

Gestion des combo boxes

Alors que vous pouvez gérer au moyen des algorithmes décrits dans la section précédente les pop-up menus, les listes déroulantes, les zones de défilement et les onglets, vous devez gérer les combo boxes différemment.

Une combo box est en réalité une zone de texte saisissable à laquelle est rattachée une liste de valeurs prédéfinies (les éléments d'un tableau). L'utilisateur peut choisir une valeur dans cette liste, et ensuite éditer le texte. En conséquence, pour une combo box, la notion d'élément sélectionné ne s'applique pas.

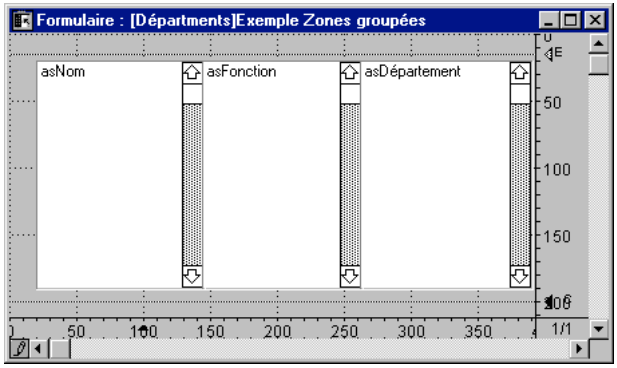
Avec les combo boxes, il n'y a jamais d'élément sélectionné. A chaque fois que l'utilisateur sélectionne une des valeurs attachées à la zone, cette valeur est placée dans l'élément zéro du tableau. Ensuite, si l'utilisateur modifie le texte, la valeur modifiée est aussi placée dans cet élément zéro. Exemple :

```
   ` Méthode objet Combo Box asCouleurs
Au cas ou
   : (Evenement formulaire=Sur chargement)
      TABLEAU ALPHA(31;asCouleurs;3)
      asCouleurs{1} :="Bleu"
      asCouleurs{2} :="Blanc"
      asCouleurs{3} :="Rouge"
   : (Evenement formulaire=Sur clic souris)
      Si (asCouleurs{0} #"")
            ` L'objet change automatiquement de valeur
            ` L'utilisation de l'événement Sur clic souris avec une Combo box
            ` n'est requise que pour prendre en compte des actions supplémentaires
      Fin de si
   : (Evenement formulaire=Sur données modifiées)
         ` Chercher dans tableau ignore l'élément 0, et donc retourne -1 ou >0
      Si (Chercher dans tableau(asCouleurs;asCouleurs{0} )<0)
            ` La valeur saisie n'est pas une des valeurs attachées à l'objet
            ` Ajouter la valeur à la liste pour la prochaine fois
         $vlElem:=Taille tableau (asCouleurs)+1
         INSERER LIGNES(asCouleurs;$vlElem)
         asCouleurs{$vlElem} :=asCouleurs{0}
      Sinon
            ` La valeur entrée est une des valeurs attachées à l'objet
      Fin de si
Fin de cas
```

Référence

Présentation des tableaux, Zones de défilement groupées.

Vous pouvez grouper des zones de défilement pour l'affichage dans un formulaire. Lorsque plusieurs zones de défilement sont groupées, elles se comportent comme une seule zone de défilement. Chaque zone peut disposer de ses propres police et taille de caractères ; cependant, nous vous recommandons d'utiliser la même hauteur de police dans chaque colonne. Lors de la saisie de données, seule la zone de défilement située au premier plan affiche une barre de défilement. Voici trois zones de défilement groupées en mode Structure, dans l'éditeur de formulaires :

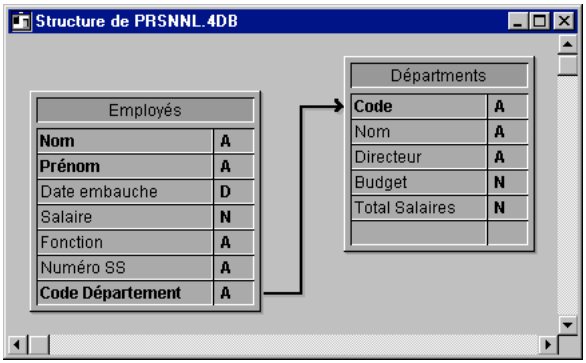


Voici quelques conseils pour la création de zones de défilement groupées :
• Assurez-vous que tous les tableaux ont la même taille (nombre d'éléments).
• Utilisez la même taille de caractères dans chaque zone.
• Veillez à ce toutes les zones aient la même hauteur.
• Alignez toutes les zones sur le haut.
• Assurez-vous que les zones ne se chevauchent pas.
• Assurez-vous que la zone de droite est au premier plan, car la barre de défilement apparaît sur la zone de premier plan.
• Groupez les zones (au moyen de la commande de menu **Grouper**) pour qu'elles fonctionnent comme s'il s'agissait d'une seule zone de défilement.

Cette méthode projet remplit les trois tableaux et les affiche à l'écran :

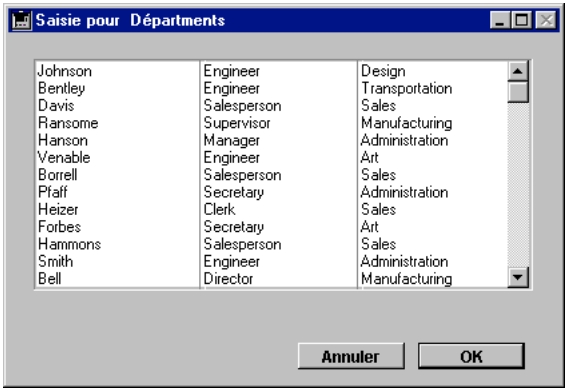
```
TOUT SELECTIONNER(Employés)
SELECTION VERS TABLEAU([Employés]Nom de Famille;asNom;[Employés]Fonction;
                                    asFonction;[Départements]Nom;asDépartement)
DIALOGUE([Départements];"Exemple Zones gorupées")
```

Cette méthode utilise les données des champs de la table [Employés] et de la table [Départements]. Voici ces tables :



Note : La table [Départements] peut être utilisée à condition qu'il y ait un lien automatique allant de [Employés] à [Départements].

Voici le résultat :

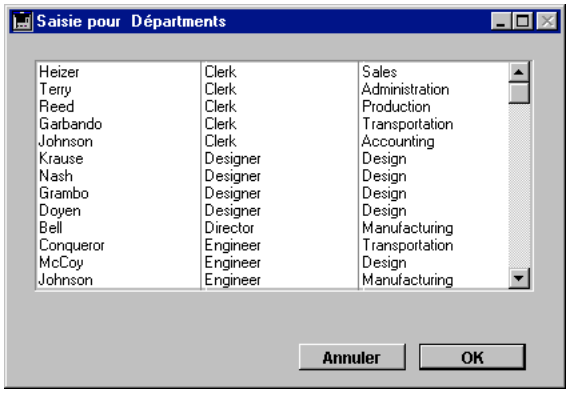


Notez qu'une seule barre de défilement est affichée. Quand l'utilisateur clique sur une ligne, les trois zones sont sélectionnées simultanément. La variable associée à chaque zone de défilement prend le numéro de la ligne ayant reçu le clic souris ; seule la méthode objet de la zone cliquée est exécutée. Par exemple, si l'utilisateur clique sur le nom “Bentley”, asNom, asFonction, et asDépartement prennent la valeur 2, mais seule la méthode objet de asNom est exécutée.

Les tableaux peuvent être triés au moyen de la commande TRIER TABLEAU. Par exemple :

TRIER TABLEAU(asFonction;asNom;asDépartement;>)

Voici le résultat du tri :



Notez que les tableaux ont été triés sur la base du premier argument de la commande TRIER TABLEAU ; les deux autres tableaux ont été désignés pour assurer la synchronisation entre les lignes. La commande TRIER TABLEAU trie toujours les tableaux (s'il y en a plusieurs) sur les valeurs du premier argument et assure la synchronisation des autres tableaux.

Note : La commande TRIER TABLEAU ne permet pas de trier les tableaux sur plusieurs niveaux. Pour afficher un tableau similaire au tableau ci-dessus tout en ayant un tri sur plusieurs niveaux (par exemple par département, puis par fonction, puis par nom), utilisez un sous-formulaire dans lequel vous affichez la table et utilisez la commande TRIER.

Référence

Présentation des tableaux, Tableaux et objets de formulaire.

Les tableaux sont des variables 4D. Comme toute variable, un tableau a une portée (une aire d'action) et suit les règles du langage 4D, à certaines différences près.

Tableaux locaux, process et interprocess

Vous pouvez créer des tableaux locaux, process ou interprocess, par exemple :

```
   ` Ceci crée un tableau local de 100 valeurs entières sur 2 octets
TABLEAU ENTIER ($aiCodes;100)
   ` Ceci crée un tableau process 100 valeurs entières sur 2 octets
TABLEAU ENTIER (aiCodes;100)
   ` Ceci crée un tableau interprocess 100 valeurs entières sur 2 octets
TABLEAU ENTIER (<>aiCodes;100)
```

La portée de ces tableaux est identique à celle des autres variables locales, process et interprocess.

Tableaux locaux

Vous déclarez un tableau local lorsque son nom commence par le signe dollar ($).

La portée d'un tableau local est la méthode dans laquelle il est créé. Le tableau est effacé lorsque la méthode est terminée. Des tableaux locaux de même nom peuvent avoir des types différents dans deux méthodes différentes, car ce sont en réalité des variables différentes n'ayant pas la même portée.

Lorsque vous créez un tableau local dans une méthode formulaire ou objet, ou dans une méthode projet appelée comme sous-routine par l'un des deux types de méthodes précédents, le tableau est créé puis effacé à chaque fois que la méthode formulaire ou la méthode objet est utilisée. En d'autres termes, le tableau est créé puis effacé pour chaque événement formulaire. Par conséquent, vous ne pouvez pas utiliser de tableaux locaux dans des formulaires, ni pour l'affichage ni pour l'impression.

Comme dans le cas des variables locales, il est préférable d'utiliser les tableaux locaux à chaque fois que c'est possible. Vous réduisez ainsi la quantité de mémoire nécessaire pour votre application.

Tableaux process

Vous déclarez un tableau process lorsque le nom du tableau débute par une simple lettre. La portée d'un tableau process est le process dans lequel il a été créé. Le tableau est effacé lorsque le process se termine ou est tué. Un tableau process dispose d'une instance créée automatiquement pour chaque process. Par conséquent, le tableau est du même type pour tous les process. En revanche, son contenu est particulier à chaque process.

Tableaux interprocess

Vous déclarez un tableau interprocess lorsque son nom commence par <> (sous Windows et MacOS) ou par le symbole diamant (sous MacOS uniquement — Option+v sur un clavier français).
La portée d'un tableau interprocess est la totalité des process pendant la session de travail. Il convient de ne les utiliser que pour partager des données ou transférer des informations entre les process.

Astuce : Quand vous savez d'avance qu'un tableau interprocess sera utilisé par plusieurs process, ce qui peut provoquer des conflits, protégez l'accès à ce tableau par un sémaphore. Pour plus d'informations, reportez-vous à l'exemple de la commande Semaphore.

Note : Vous pouvez utiliser des tableaux process et interprocess dans des formulaires pour créer des objets de formulaire tels que des zones de défilement, des listes déroulantes, etc.

Passer un tableau comme paramètre

Vous pouvez passer un tableau en tant que paramètre à une commande 4D ou à une routine d'un plug-in 4D. Cependant, vous ne pouvez pas passer un tableau comme paramètre à une méthode utilisateur. La solution dans ce cas est de passer un pointeur vers le tableau comme paramètre à cette méthode. Pour plus de détails, reportez-vous à la section Tableaux et pointeurs.

Affecter un tableau à un autre tableau

Contrairement à ce que vous pouvez faire avec des variables de type Texte ou Chaîne, vous ne pouvez pas affecter un tableau à un autre tableau. Pour copier (affecter) un tableau à un autre, utilisez la fonction COPIER TABLEAU.

Référence

Présentation des tableaux, Tableaux et pointeurs.

Vous pouvez passer un tableau comme paramètre à une commande 4D ou à une routine d'un Plug-in 4D. Cependant, vous ne pouvez pas passer un tableau comme paramètre dans une méthode utilisateur. La solution consiste à passer un pointeur vers le tableau comme paramètre de la méthode.

Note : Vous pouvez passer des tableaux process et des tableaux interprocess comme paramètres, mais pas des tableaux locaux.

Voici quelques exemples.

• Prenons le cas suivant :

```
Si ((0<atNoms)&(atNoms<Taille tableau(atNoms))
      ` Si possible, sélectionner l'élément suivant l'élément sélectionné
   atNoms:=atNoms+1
Fin de si
```

Si vous avez besoin de faire la même chose pour 50 tableaux différents, vous pouvez vous éviter d'écrire 50 fois la même chose, en utilisant la méthode projet suivante :

```
   ` Méthode projet SELECTIONNER ELEMENT SUIVANT
   ` SELECTIONNER ELEMENT SUIVANT ( Pointeur )
   ` SELECTIONNER ELEMENT SUIVANT ( -> Tableau )

C_POINTEUR ($1)

Si ((0<$1->)&($1-><Taille tableau($1->))
   $1->:=$1->+1 ` Si possible, sélectionner l'élément suivant l'élément sélectionné
Fin de si
```

Ensuite, vous pouvez écrire :

```
SELECTIONNER ELEMENT SUIVANT (->atNoms)
   ` ...
SELECTIONNER ELEMENT SUIVANT (->asCodesPostaux)
   ` ...
SELECTIONNER ELEMENT SUIVANT (->alEnrgsIDs)
   ` ... et ainsi de suite.
```

• La méthode projet suivante retourne la somme de tous les éléments d'un tableau numérique (Entier, Entier long, ou Réel) :

```
` Somme Tableau
` Somme Tableau ( Pointeur )
` Somme Tableau ( -> Tableau )

C_REEL ($0)
$0:=0
Boucle ($vlElem;1;Taille tableau($1->))
   $0:=$0+$1->{$vlElem}
Fin de boucle
```

Ensuite, vous pouvez écrire :

```
$vlSomme:=Somme Tableau (->arSalaires)
   ` ...
$vlSomme:=Somme Tableau (->aiDefectCounts)
   ` ..
$vlSomme:=Somme Tableau (->alPopulations)
```

• Cette méthode met une majuscule à tous les éléments d'un tableau Alpha ou Texte :

```
` MAJUSCULE TABLEAU
` MAJUSCULE TABLEAU ( Pointeur )
` MAJUSCULE TABLEAU ( -> Tableau )

Boucle ($vlElem;1;Taille tableau($1->))
   Si ($1->{$vlElem} #"")
      $1->{$vlElem} :=Majusc($1->{$vlElem} [[1]])+
                                                Minusc(Sous chaine($1->{$vlElem} ;2))
   Fin de si
Fin de boucle
```

Ensuite, vous pouvez écrire :

```
MAJUSCULE TABLEAU (->atSujets )
   ` ...
MAJUSCULE TABLEAU (->asNomsFamille )
```

La combinaison de tableaux, pointeurs et boucles telles que Boucle...Fin de boucle vous permet d'écrire un grand nombre de petites méthodes projet très utiles pour gérer les tableaux.

Référence

Les tableaux et le langage 4D, Présentation des tableaux.

Un tableau a toujours un élément zéro. Même si l'élément zéro n'est pas affiché lorsqu'un tableau est utilisé pour remplir un objet de formulaire, vous pouvez l'utiliser sans réserve dans le langage.

Un exemple possible d'utilisation de l'élément zéro est le cas de la combo box examiné dans la section Tableaux et objets de formulaire.

Voici deux autres exemples :

(1) Si vous voulez exécuter une action seulement lorsque vous cliquez sur un élément autre que l'élément préalablement sélectionné, vous pouvez garder la trace de chaque élément sélectionné. Une façon de le faire est d'utiliser une variable process dans laquelle vous conservez le numéro de l'élément sélectionné. Une autre manière consiste à utiliser l'élément zéro du tableau :

```
   ` Méthode objet zone de défilement atNoms
Au cas ou
   : (Evenement formulaire=Sur chargement)
         ` Initialisons le tableau
      TABLEAU TEXTE (atNoms;5)
         ` ...
         ` Initialiser l'élément zéro avec le numéro
         ` de l'élément courant sélectionné sous sa forme alphanumérique
         ` Ici vous commencez sans élément sélectionné
      atNoms{0} :="0"

   : (Evenement formulaire=Sur libération)
         ` Nous n'avons plus besoin du tableau
      EFFACER VARIABLE(atNoms)

   : (Evenement formulaire=Sur clic souris)
      Si (atNoms#0)
         Si (atNoms#Num(atNoms{0} ))
            vtInfo:="Vous avez cliqué sur : "+atNoms{atNoms} +" qui n'était pas
                                                      précédemment sélectionné."
            atNoms{0} :=Chaine(atNoms)
         Fin de si
      Fin de si
```

```
    : (Evenement formulaire=Sur doucle clic souris)
        Si (atNoms#0)
            ALERTE ("Vous avez double-cliqué sur : "+atNoms{atNoms})
        Fin de si
Fin de cas
```

(2) Lorsque 4D envoie des caractères vers un document ou le port série, ou bien en reçoit, vous avez la possibilité de filtrer les codes ASCII entre les plates-formes et systèmes dont les tables ASCII diffèrent — les commandes UTILISER FILTRE, Mac vers ISO, ISO vers Mac, Mac vers Windows et Windows vers Mac.

Dans certains cas, vous pouvez souhaiter contrôler intégralement la traduction des codes ASCII. Pour cela, vous pouvez utiliser un tableau d'entiers de 255 éléments, dans lequel le Nième élément est la traduction ASCII du caractère dont le code ASCII d'origine est N. Par exemple, si le code ASCII 187 doit devenir 156, vous écrivez <>tiFiltreAsciiExport{187}:=156 et <>tiFiltreAsciiImport{156}:=187 dans la méthode qui initialise les tableaux interprocess utilisés dans la base. Vous pouvez alors envoyer une suite de caractères en utilisant une méthode projet telle que celle-ci :

```
    ` X ENVOYER PAQUET ( Texte { ; Heure }  )
Boucle ($vlChar;1;Longueur($1))
    $1[[vlChar]]:=Caractere(<>tiFiltreAsciiExport{Code ascii($1[[vlChar]])} )
Fin de boucle
Si (Nombre de parametres>=2)
    ENVOYER PAQUET ($2;$1)
Sinon
    ENVOYER PAQUET ($1)
Fin de si

    ` X Recevoir paquet ( Texte { ; Heure }  ) -> Texte
Si (Nombre de parametres>=2)
    RECEVOIR PAQUET ($2;$1)
Sinon
    RECEVOIR PAQUET ($1)
Fin de si
$0:=$1
Boucle ($vlChar;1;Longueur($1))
    $0[[vlChar]]:=Caractere(<>tiFiltreAsciiImport{Code ascii($0[[vlChar]])} )
Fin de boucle
```

Dans cet exemple, si une suite de caractères contenant des caractères NULL (code ASCII zéro) est envoyée ou reçue, l'élément zéro des tableaux <>tiFiltreAsciiExport et <>tiFiltreAsciiImport jouera son rôle comme n'importe lequel des 255 éléments du tableau.

Tableaux à deux dimensions

Chaque commande de déclaration de tableau permet de créer ou de redimensionner des tableaux à une ou à deux dimensions. Exemple :

```
   ` Créer un tableau texte composé de 100 lignes de 50 colonnes
TABLEAU TEXTE (atTopics;100;50)
```

Les tableaux à deux dimensions sont essentiellement des objets de langage ; vous ne pouvez ni les afficher ni les imprimer.

Dans l'exemple prédédent :
- atTopics est un tableau à deux dimensions.
- atTopics{8} {5} est le 5e élément (5e colonne...) de la 8e ligne.
- atTopics{20} est la 20e ligne et est elle-même un tableau à une dimension.
- Taille tableau(atTopics) retourne 100, qui est le nombre de lignes
- Taille tableau(atTopics{17}) retourne 50, qui est le nombre de colonnes de la 17e ligne

Dans l'exemple suivant, un pointeur vers chaque champ de chaque table de la base est stocké dans un tableau à deux dimensions :

```
   ` Créer autant de lignes vides qu'il y a de tables
TABLEAU POINTEUR (<>apChamps;Nombre de tables;0)
   ` Pour chaque table
Boucle ($vlTable;1;Taille tableau(<>apChamps))
      ` Redimensionner la ligne avec autant de colonnes qu'il y a de champs
      ` dans la table
   INSERER LIGNES (<>apFields{$vlTable} ;1;Nombre de champs($vlTable))
      ` Donner la valeur des éléments
   Boucle ($vlChamp;1;Taille tableau(<>apChamps{$vlTable} ))
      <>apChamps{$vlTable} {$vlChamp} :=Champ($vlTable;$vlChamp)
   Fin de boucle
Fin de boucle
```

Dans la mesure où le tableau à deux dimensions a été initialisé, vous pouvez obtenir ainsi les pointeurs vers les champs d'une table de votre choix :

```
   ` Obtenir les pointeurs vers les champs pour la table affichée à l'écran:
COPIER TABLEAU (<>apChamps{Table(Table du formulaire courant)};
                                                $apMesChampsdeTravail)
   ` Initialiser les champs booléens et date
Boucle ($vlElem;1;Taille tableau($apMesChampsdeTravail))
   Au cas ou
      : (Type($apMesChampsdeTravail{$vlElem} ->)=Is Date)
         $apMesChampsdeTravail{$vlElem} ->:=Date du jour
      : (Type($apMesChampsdeTravail{$vlElem} ->)=Is Boolean)
         $apMesChampsdeTravail{$vlElem} ->:=Vrai
   Fin de cas
Fin de boucle
```

Note : Comme le montre cet exemple, les lignes des tableaux à deux dimensions peuvent être ou non de la même taille, indifféremment.

Référence

Présentation des tableaux.

A la différence des données que vous stockez sur disque lorsque vous utilisez des tables ou des enregistrements, un tableau **réside toujours en mémoire dans son intégralité**.

Par exemple, si tous les codes postaux américains étaient saisis dans une table [Codes Postaux], celle-ci contiendrait environ 100 000 enregistrements. De plus, cette table comporterait plusieurs champs : le code postal lui-même ainsi que la ville, le comté et l'état correspondants. Si vous ne sélectionnez que les codes postaux de Californie, 4D crée la sélection d'enregistrements correspondante à l'intérieur de la table [Codes Postaux], et ensuite ne charge les enregistrements que lorsque vous en avez besoin (par exemple, pour les afficher ou les imprimer). En d'autres termes, vous travaillez avec une série ordonnée de valeurs (du même type pour chaque champ) partiellement chargée du disque en mémoire.

Procéder de la même manière avec les tableaux serait laborieux, pour les raisons suivantes :

• Pour maintenir les quatre types d'information (code postal, ville, comté, état), vous auriez besoin de quatre grands tableaux en mémoire.
• Comme un tableau réside en mémoire dans son intégralité, vous seriez obligé de garder tous les codes postaux en mémoire pendant toute la session de travail, même si les données n'étaient pas utilisées en permanence.
• Toujours parce qu'un tableau réside en mémoire dans son intégralité, les quatre tableaux devraient être chargés ou sauvegardés sur le disque à chaque fois que vous démarreriez ou quitteriez l'application, quand bien même les données ne seraient d'aucune utilité pour la session de travail.

Conclusion : Les tableaux ont pour rôle de manipuler une certaine quantité de données pendant une période brève. En contrepartie, comme ils résident en mémoire, ils sont d'une utilisation rapide et facile.

Cependant, dans certaines circonstances, vous pouvez avoir besoin de tableaux contenant des centaines ou des milliers d'éléments. Voici les formules à appliquer pour calculer la quantité de mémoire utilisée pour chaque type de tableau:

Type de Tableau	Calcul de la quantité de mémoire en octets
Booléen	(31+nombre d'éléments)/8
Date	(1+nombre d'éléments) * 6
Alpha	(1+nombre d'éléments) * longueur déclarée (+1 si impair, +2 si pair)
Entier	(1+nombre d'éléments) * 2
Entier long	(1+nombre d'éléments) * 4
Image	(1+nombre d'éléments) * 4 + somme de la taille de chaque image
Pointeur	(1+nombre d'éléments) * 16
Réel	(1+nombre d'éléments) * 8 (Windows, PPC) ou * 10 (68K)
Texte	(1+nombre d'éléments) * 6 + somme de la taille de chaque texte
Deux dimensions	(1+nombre d'éléments) * 12 + somme de la taille de chaque tableau

Note : Quelques octets supplémentaires sont requis pour le repérage de l'élément, le nombre d'éléments et le tableau lui-même.

Lorsque vous travaillez avec de très grands tableaux, la meilleure façon de gérer d'éventuels problèmes de saturation de la mémoire est d'accompagner la création de tableau d'une méthode projet APPELER SUR ERREUR. Exemple :

```
    ` Vous allez lancer une opération batch fonctionnant toute la nuit
    ` qui requiert la création de grands tableaux. Pour éviter
    ` des erreurs en pleine nuit, créez les tableaux au début de
    ` l'opération et testez les erreurs au même moment :
gError:=0 ` Initialisation
    ` Installation de la méthode de gestion d'erreurs
APPELER SUR ERREUR ("GESTION ERREUR")
TABLEAU ALPHA (63;asCeTableau;50000) ` Environ 3125 Ko
TABLEAU REEL (arCetAutreTableau;50000) ` 488 Ko
APPELER SUR ERREUR ("") ` Il n'est plus nécessaire d'intercepter les erreurs
Si (gError=0)
    ` Les tableaux ont pu être créés
    ` poursuivons l'opération
Sinon
   ALERTE ("Cette opération requiert davantage de mémoire !")
Fin de si
    ` Dans tous les cas, nous n'avons plus besoin des tableaux
EFFACER VARIABLE (asCeTableau)
EFFACER VARIABLE (arCetAutreTableau)
```

La méthode projet GESTION ERREUR est la suivante :

```
` Méthode projet GESTION ERREUR
gError:=Error  ` Retourner le code d'erreur
```

Référence

APPELER SUR ERREUR, Présentation des tableaux.

TABLEAU ENTIER (nomTableau; taille{; taille2})

Paramètre	Type		Description
nomTableau	Tableau	→	Nom du tableau
taille	Numérique	→	Nombre d'éléments du tableau ou Nombre de tableaux si taille2 est spécifié
taille2	Numérique	→	Nombre d'éléments des tableaux à deux dimensions

Description

La commande TABLEAU ENTIER crée et/ou redimensionne un tableau d'éléments de type Entier (2 octets) en mémoire.

• Le paramètre nomTableau est le nom du tableau.
• Le paramètre taille est le nombre d'éléments du tableau.
• Le paramètre taille2 est optionnel. Si vous le spécifiez, cette commande crée un tableau à deux dimensions. Dans ce cas, taille spécifie le nombre de lignes et taille2 spécifie le nombre de colonnes de chaque tableau. Chaque ligne dans un tableau à deux dimensions peut être traitée à la fois comme un élément et comme un tableau. Cela signifie que vous pouvez insérer et supprimer des tableaux entiers dans un tableau à deux dimensions, par l'intermédiaire des autres commandes de ce thème, lorsque vous travaillez avec la première dimension du tableau.

Lorsque vous appliquez la commande TABLEAU ENTIER à un tableau existant :
• Si vous agrandissez sa taille, les éléments existants ne sont pas modifiés, les nouveaux éléments sont initialisés à 0.
• Si vous réduisez sa taille, les éléments du "bas" du tableau sont supprimés et perdus.

Exemples

(1) Cet exemple crée un tableau process contenant 100 éléments de type Entier :

⇒ **TABLEAU ENTIER** (tabEntiers; 100)

(2) Cet exemple crée un tableau local de 100 lignes contenant chacune 50 éléments de type Entier :

⇒ **TABLEAU ENTIER** ($tabEntiers;100;50)

(3) Cet exemple crée un tableau interprocess de 50 éléments de type Entier et affecte à chaque élément son numéro :

```
⇒    TABLEAU ENTIER (<>tabEntiers; 50)
       Boucle($vElem;1;50)
         <>tabEntiers{$vElem}:=$vElem
       Fin de boucle
```

Référence

TABLEAU ENTIER LONG, TABLEAU REEL.

TABLEAU ENTIER LONG

version 5

TABLEAU ENTIER LONG (nomTableau; taille{; taille2})

Paramètre	Type		Description
nomTableau	Tableau	→	Nom du tableau
taille	Numérique	→	Nombre d'éléments du tableau ou Nombre de tableaux si taille2 est spécifié
taille2	Numérique	→	Nombre d'éléments des tableaux à deux dimensions

Description

La commande TABLEAU ENTIER LONG crée et/ou redimensionne un tableau d'éléments de type Entier long (4 octets) en mémoire.

• Le paramètre nomTableau est le nom du tableau.
• Le paramètre taille est le nombre d'éléments du tableau.
• Le paramètre taille2 est optionnel. Si vous le spécifiez, cette commande crée un tableau à deux dimensions. Dans ce cas, taille spécifie le nombre de lignes et taille2 spécifie le nombre de colonnes de chaque tableau. Chaque ligne dans un tableau à deux dimensions peut être traitée à la fois comme un élément et comme un tableau. Cela signifie que vous pouvez insérer et supprimer des tableaux entiers dans un tableau à deux dimensions, par l'intermédiaire des autres commandes de ce thème, lorsque vous travaillez avec la première dimension du tableau.

Lorsque vous appliquez la commande TABLEAU ENTIER LONG à un tableau existant :
• Si vous agrandissez sa taille, les éléments existants ne sont pas modifiés, les nouveaux éléments sont initialisés à 0.
• Si vous réduisez sa taille, les éléments du "bas" du tableau sont supprimés et perdus.

Exemples

(1) Cet exemple crée un tableau process contenant 100 éléments de type Entier long :

⇒ **TABLEAU ENTIER LONG** (tabEntiersLongs; 100)

(2) Cet exemple crée un tableau local de 100 lignes contenant chacune 50 éléments de type Entier long :

⇒ **TABLEAU ENTIER LONG** ($tabEntiersLongs;100;50)

(3) Cet exemple crée un tableau interprocess de 50 éléments de type Entier long et affecte à chaque élément son numéro :

```
⇒    TABLEAU ENTIER LONG (<>tabEntiersLongs; 50)
       Boucle($vElem;1;50)
         <>tabEntiersLongs{$vElem}:=$vElem
       Fin de boucle
```

Référence

TABLEAU ENTIER, TABLEAU REEL.

TABLEAU REEL (nomTableau; taille{; taille2})

Paramètre	Type		Description
nomTableau	Tableau	→	Nom du tableau
taille	Numérique	→	Nombre d'éléments du tableau ou Nombre de tableaux si taille2 est spécifié
taille2	Numérique	→	Nombre d'éléments des tableaux à deux dimensions

Description

La commande TABLEAU REEL crée et/ou redimensionne un tableau d'éléments de type Réel (numérique) en mémoire.

• Le paramètre nomTableau est le nom du tableau.
• Le paramètre taille est le nombre d'éléments du tableau.
• Le paramètre taille2 est optionnel. Si vous le spécifiez, cette commande crée un tableau à deux dimensions. Dans ce cas, taille spécifie le nombre de lignes et taille2 spécifie le nombre de colonnes de chaque tableau. Chaque ligne dans un tableau à deux dimensions peut être traitée à la fois comme un élément et comme un tableau. Cela signifie que vous pouvez insérer et supprimer des tableaux entiers dans un tableau à deux dimensions, par l'intermédiaire des autres commandes de ce thème, lorsque vous travaillez avec la première dimension du tableau.

Lorsque vous appliquez la commande TABLEAU REEL à un tableau existant :
• Si vous agrandissez sa taille, les éléments existants ne sont pas modifiés, les nouveaux éléments sont initialisés à 0.
• Si vous réduisez sa taille, les éléments du "bas" du tableau sont supprimés et perdus.

Exemples

(1) Cet exemple crée un tableau process contenant 100 éléments de type Réel :

⇒ **TABLEAU REEL** (tabRéel; 100)

(2) Cet exemple crée un tableau local de 100 lignes contenant chacune 50 éléments de type Réel :

⇒ **TABLEAU REEL** ($tabRéel; 100;50)

(3) Cet exemple crée un tableau interprocess de 50 éléments de type Réel et affecte à chaque élément son numéro :

```
⇒    TABLEAU REEL (<>tabRéel; 50)
       Boucle($vElem;1;50)
         <>tabRéel{$vElem}:=$vElem
       Fin de boucle
```

Référence

TABLEAU ENTIER, TABLEAU ENTIER LONG.

TABLEAU ALPHA (longueurChaîne; nomTableau; taille{; taille2})

Paramètre	Type		Description
longueurChaîne	Numérique	→	Longueur de la chaîne (1..255)
nomTableau	Tableau	→	Nom du tableau
taille	Numérique	→	Nombre d'éléments du tableau ou Nombre de tableaux si taille2 est spécifié
taille2	Numérique	→	Nombre d'éléments des tableaux à deux dimensions

Description

La commande TABLEAU ALPHA crée et/ou redimensionne un tableau d'éléments de type Chaîne alphanumérique en mémoire.

• Le paramètre longueurChaîne définit le nombre maximum de caractères que chaque élément du tableau peut contenir. Cette longueur doit être comprise entre 1 et 255 caractères.
• Le paramètre nomTableau est le nom du tableau.
• Le paramètre taille est le nombre d'éléments du tableau.
• Le paramètre taille2 est optionnel. Si vous le spécifiez, cette commande crée un tableau à deux dimensions. Dans ce cas, taille spécifie le nombre de lignes et taille2 spécifie le nombre de colonnes de chaque tableau. Chaque ligne dans un tableau à deux dimensions peut être traitée à la fois comme un élément et comme un tableau. Cela signifie que vous pouvez insérer et supprimer des tableaux entiers dans un tableau à deux dimensions, par l'intermédiaire des autres commandes de ce thème, lorsque vous travaillez avec la première dimension du tableau.

Lorsque vous appliquez la commande TABLEAU ALPHA à un tableau existant :
• Si vous agrandissez sa taille, les éléments existants ne sont pas modifiés, les nouveaux éléments sont initialisés à "" (chaîne vide).
• Si vous réduisez sa taille, les éléments du "bas" du tableau sont supprimés et perdus.

Exemples

(1) Cet exemple crée un tableau process comportant 100 éléments. Chaque élément est une chaîne de 31 caractères :

⇒ **TABLEAU ALPHA** (31;tabAlpha;100)

(2) Cet exemple crée un tableau local de 100 lignes. Chaque ligne comporte 50 éléments, chaque élément est une chaîne de 63 caractères :

⇒ **TABLEAU ALPHA** (63;$tabAlpha;100;50)

(3) Cet exemple crée un tableau interprocess comportant 50 éléments. Chaque élément est une chaîne de 255 caractères. La valeur "Elément No" suivie du numéro de l'élément est affectée à chaque élément :

```
⇒ TABLEAU ALPHA (255;<>tabAlpha;50)
  Boucle ($vlElem;1;50)
     <>tabAlpha{$vlElem}:="Elément No" + Chaine($vlElem)
  Fin de boucle
```

Référence

TABLEAU TEXTE.

TABLEAU TEXTE (nomTableau; taille{; taille2})

Paramètre	Type		Description
nomTableau	Tableau	→	Nom du tableau
taille	Numérique	→	Nombre d'éléments du tableau ou Nombre de tableaux si taille2 est spécifié
taille2	Numérique	→	Nombre d'éléments des tableaux à deux dimensions

Description

La commande TABLEAU TEXTE crée et/ou redimensionne un tableau d'éléments de type Texte en mémoire.

• Le paramètre nomTableau est le nom du tableau.
• Le paramètre taille est le nombre d'éléments du tableau.
• Le paramètre taille2 est optionnel. Si vous le spécifiez, cette commande crée un tableau à deux dimensions. Dans ce cas, taille spécifie le nombre de lignes et taille2 spécifie le nombre de colonnes de chaque tableau. Chaque ligne dans un tableau à deux dimensions peut être traitée à la fois comme un élément et comme un tableau. Cela signifie que vous pouvez insérer et supprimer des tableaux entiers dans un tableau à deux dimensions, par l'intermédiaire des autres commandes de ce thème, lorsque vous travaillez avec la première dimension du tableau.

Lorsque vous appliquez la commande TABLEAU TEXTE à un tableau existant :
• Si vous agrandissez sa taille, les éléments existants ne sont pas modifiés, les nouveaux éléments sont initialisés à "" (chaîne vide).
• Si vous réduisez sa taille, les éléments du "bas" du tableau sont supprimés et perdus.

Exemples

(1) Cet exemple crée un tableau process contenant 100 éléments de type Texte :

```
⇒ TABLEAU TEXTE (tabTexte; 100)
```

(2) Cet exemple crée un tableau local de 100 lignes contenant chacune 50 éléments de type Texte :

```
⇒ TABLEAU TEXTE ($tabEntiersLongs;100;50)
```

(3) Cet exemple crée un tableau interprocess de 50 éléments de type Texte et affecte à chaque élément la valeur "Elément No" suivie du numéro de l'élément :

```
⇒    TABLEAU TEXTE (<>tabTexte; 50)
       Boucle($vElem;1;50)
         <>tabTexte{$vElem}:="Elément n°"+ Chaine ($vElem)
       Fin de boucle
```

Référence

TABLEAU ALPHA.

TABLEAU DATE (nomTableau; taille{; taille2})

Paramètre	Type		Description
nomTableau	Tableau	→	Nom du tableau
taille	Numérique	→	Nombre d'éléments du tableau ou Nombre de tableaux si taille2 est spécifié
taille2	Numérique	→	Nombre d'éléments des tableaux à deux dimensions

Description

La commande TABLEAU DATE crée et/ou redimensionne un tableau d'éléments de type Date en mémoire.

• Le paramètre nomTableau est le nom du tableau.
• Le paramètre taille est le nombre d'éléments du tableau.
• Le paramètre taille2 est optionnel. Si vous le spécifiez, cette commande crée un tableau à deux dimensions. Dans ce cas, taille spécifie le nombre de lignes et taille2 spécifie le nombre de colonnes de chaque tableau. Chaque ligne dans un tableau à deux dimensions peut être traitée à la fois comme un élément et comme un tableau. Cela signifie que vous pouvez insérer et supprimer des tableaux entiers dans un tableau à deux dimensions, par l'intermédiaire des autres commandes de ce thème, lorsque vous travaillez avec la première dimension du tableau.

Lorsque vous appliquez la commande TABLEAU DATE à un tableau existant :
• Si vous agrandissez sa taille, les éléments existants ne sont pas modifiés, les nouveaux éléments sont initialisés à la valeur de date nulle (!00/00/00!).
• Si vous réduisez sa taille, les éléments du "bas" du tableau sont supprimés et perdus.

Exemples

(1) Cet exemple crée un tableau process contenant 100 éléments de type Date :

```
⇒ TABLEAU DATE (tabDates; 100)
```

(2) Cet exemple crée un tableau local de 100 lignes contenant chacune 50 éléments de type Date :

```
⇒ TABLEAU DATE ($tabDates;100;50)
```

(3) Cet exemple crée un tableau interprocess de 50 éléments de type Date et affecte à chaque élément la date du jour + un nombre de jours égal au numéro de l'élément :

```
⇒    TABLEAU DATE (<>tabDates;50)
       Boucle($vElem;1;50)
         <>tabDates{$vElem}:=Date du jour+$vElem
       Fin de boucle
```

TABLEAU BOOLEEN (nomTableau; taille{; taille2})

Paramètre	Type		Description
nomTableau	Tableau	→	Nom du tableau
taille	Numérique	→	Nombre d'éléments du tableau ou Nombre de tableaux si taille2 est spécifié
taille2	Numérique	→	Nombre d'éléments des tableaux à deux dimensions

Description

La commande TABLEAU BOOLEEN crée et/ou redimensionne un tableau d'éléments de type Booléen en mémoire.

• Le paramètre nomTableau est le nom du tableau.
• Le paramètre taille est le nombre d'éléments du tableau.
• Le paramètre taille2 est optionnel. Si vous le spécifiez, cette commande crée un tableau à deux dimensions. Dans ce cas, taille spécifie le nombre de lignes et taille2 spécifie le nombre de colonnes de chaque tableau. Chaque ligne dans un tableau à deux dimensions peut être traitée à la fois comme un élément et comme un tableau. Cela signifie que vous pouvez insérer et supprimer des tableaux entiers dans un tableau à deux dimensions, par l'intermédiaire des autres commandes de ce thème, lorsque vous travaillez avec la première dimension du tableau.

Lorsque vous appliquez la commande TABLEAU BOOLEEN à un tableau existant :
• Si vous agrandissez sa taille, les éléments existants ne sont pas modifiés, les nouveaux éléments sont initialisés à Faux.
• Si vous réduisez sa taille, les éléments du "bas" du tableau sont supprimés et perdus.

Astuce : Dans certaines circonstances, l'utilisation d'un tableau d'Entiers dans lequel chaque élément différent de zéro signifie "vrai" et chaque élément égal à zéro signifie "faux" est une alternative à l'utilisation d'un tableau de Booléens.

Exemples

(1) Cet exemple crée un tableau process contenant 100 éléments de type Booléen :

⇒ **TABLEAU BOOLEEN** (tabBooléens; 100)

(2) Cet exemple crée un tableau local de 100 lignes contenant chacune 50 éléments de type Booléen :

⇒ **TABLEAU BOOLEEN** ($tabBooléens;100;50)

(3) Cet exemple crée un tableau interprocess de 50 éléments de type Booléen et affecte à chaque élément pair la valeur Faux :

```
⇒ TABLEAU BOOLEEN (<>tabBooléens;50)
  Boucle($vElem;1;50)
    <>tabBooléens{$vElem}:=(($vElem%2)=0)
  Fin de boucle
```

Référence

TABLEAU ENTIER.

TABLEAU IMAGE (nomTableau; taille{; taille2})

Paramètre	Type		Description
nomTableau	Tableau	→	Nom du tableau
taille	Numérique	→	Nombre d'éléments du tableau ou Nombre de tableaux si taille2 est spécifié
taille2	Numérique	→	Nombre d'éléments des tableaux à deux dimensions

Description

La commande TABLEAU IMAGE crée et/ou redimensionne un tableau d'éléments de type Image en mémoire.

• Le paramètre nomTableau est le nom du tableau.
• Le paramètre taille est le nombre d'éléments du tableau.
• Le paramètre taille2 est optionnel. Si vous le spécifiez, cette commande crée un tableau à deux dimensions. Dans ce cas, taille spécifie le nombre de lignes et taille2 spécifie le nombre de colonnes de chaque tableau. Chaque ligne dans un tableau à deux dimensions peut être traitée à la fois comme un élément et comme un tableau. Cela signifie que vous pouvez insérer et supprimer des tableaux entiers dans un tableau à deux dimensions, par l'intermédiaire des autres commandes de ce thème, lorsque vous travaillez avec la première dimension du tableau.

Lorsque vous appliquez la commande TABLEAU IMAGE à un tableau existant :
• Si vous agrandissez sa taille, les éléments existants ne sont pas modifiés, les nouveaux éléments sont initialisés à une image vide (ce qui signifie que la fonction Taille image appliquée à l'un de ces éléments retourne 0).
• Si vous réduisez sa taille, les éléments du "bas" du tableau sont supprimés et perdus.

Exemples

(1) Cet exemple crée un tableau process contenant 100 éléments de type Image :

```
⇒ TABLEAU IMAGE (tabImages; 100)
```

(2) Cet exemple crée un tableau local de 100 lignes contenant chacune 50 éléments de type Image :

```
⇒ TABLEAU IMAGE ($tabImages;100;50)
```

(3) Cet exemple crée un tableau interprocess d'éléments de type Image. La taille du tableau est égale au nombre de ressources 'PICT' dont le nom commence par "Utilisateur Intf/" disponibles dans la base. Chaque image est chargée dans un élément du tableau :

```
      LISTE RESSOURCES ("PICT";$aiResIDs;$asResNoms)
⇒     TABLEAU IMAGE (<>tabImages;Taille tableau($aiResIDs))
      $vlPictElem:=0
      Boucle ($vlElem;1;Taille tableau(<>tabImages))
         Si ($asResNoms="Utilisateur Intf/@")
            $vlPictElem:=vlPictElem+1
            LIRE RESSOURCE IMAGE("PICT";$aiResIDs{$vlElem};$vgImage)
            <>tabImages{$vlPictElem}:=$vgImage
         Fin de si
      Fin de boucle
⇒     TABLEAU IMAGE (<>tabImages;$vlPictElem)
```

TABLEAU POINTEUR

version 5

TABLEAU POINTEUR (nomTableau; taille{; taille2})

Paramètre	Type		Description
nomTableau	Tableau	→	Nom du tableau
taille	Numérique	→	Nombre d'éléments du tableau ou Nombre de tableaux si taille2 est spécifié
taille2	Numérique	→	Nombre d'éléments des tableaux à deux dimensions

Description

La commande TABLEAU POINTEUR crée ou redimensionne un tableau d'éléments de type Pointeur en mémoire.

• Le paramètre nomTableau est le nom du tableau.
• Le paramètre taille est le nombre d'éléments du tableau.
• Le paramètre taille2 est optionnel. Si vous le spécifiez, cette commande crée un tableau à deux dimensions. Dans ce cas, taille spécifie le nombre de lignes et taille2 spécifie le nombre de colonnes de chaque tableau. Chaque ligne dans un tableau à deux dimensions peut être traitée à la fois comme un élément et comme un tableau. Cela signifie que vous pouvez insérer et supprimer des tableaux entiers dans un tableau à deux dimensions, par l'intermédiaire des autres commandes de ce thème, lorsque vous travaillez avec la première dimension du tableau.

Lorsque vous appliquez la commande TABLEAU POINTEUR à un tableau existant :
• Si vous agrandissez sa taille, les éléments existants ne sont pas modifiés, les nouveaux éléments sont initialisés à un pointeur nul (ce qui signifie que la fonction Nil appliquée à l'un de ces éléments retourne Vrai).
• Si vous réduisez sa taille, les éléments du "bas" du tableau sont supprimés et perdus.

Exemples

(1) Cet exemple crée un tableau process contenant 100 éléments de type Pointeur :

⇒ **TABLEAU POINTEUR** (tabPointeurs; 100)

(2) Cet exemple crée un tableau local de 100 lignes contenant chacune 50 éléments de type Pointeur :

⇒ **TABLEAU POINTEUR** ($tabPointeurs;100;50)

(3) Cet exemple crée un tableau interprocess d'éléments de type Pointeur dont la taille est égale au nombre de tables dans la base et remplit chaque élément pointant vers la table dont le numéro est le même que celui de l'élément :

```
⇒    TABLEAU POINTEUR (<>tabPointeurs;Nombre de tables)
       Boucle($vElem;1;Taille tableau(<>tabPointeurs))
         <>tabPointeurs{$vElem}:=Table($vElem)
     Fin de boucle
```

Taille tableau

version 5

Taille tableau (tableau) → Numérique

Paramètre	Type		Description
tableau	Tableau	→	Tableau dont vous désirez connaître la taille
Résultat	Numérique	←	Nombre d'éléments dans le tableau

Description

Taille tableau retourne le nombre d'éléments de tableau.

Exemples

(1) L'exemple suivant retourne la taille du tableau monTableau :

```
⇒ vTaille := Taille tableau (monTableau)  ` vTaille reçoit la taille de monTableau
```

(2) L'exemple suivant retourne le nombre de lignes d'un tableau à deux dimensions :

```
⇒ vlLignes:=Taille tableau(t2DTableau)  ` vlLignes reçoit la taille de t2DTableau
```

(3) L'exemple suivant retourne le nombre de colonnes d'une ligne d'un tableau à deux dimensions :

```
   ` vlColonnes reçoit la taille de t2DTableau{10}
⇒ vlColonnes:=Taille tableau(t2DTableau{10})
```

Référence

INSERER LIGNES, SUPPRIMER LIGNES.

TRIER TABLEAU (tableau{; tableau2; ...; tableauN}{; sensDuTri})

Paramètre	Type		Description
tableau	Tableau	→	Tableau(x) à trier
sensDuTri	> ou <	→	> pour effectuer un tri par ordre croissant ou < pour effectuer un tri par ordre décroissant Tri croissant si ce paramètre est omis

Description

La commande TRIER TABLEAU trie un ou plusieurs tableaux par ordre croissant ou décroissant.

Note : Vous ne pouvez pas trier de tableaux de type Pointeur ou Image. Vous pouvez trier un élément d'un tableau à deux dimensions (c'est-à-dire t2DTableau{$vlCetElément}), mais vous ne pouvez pas trier le tableau 2D lui-même (c'est-à-dire t2DTableau).

Le paramètre sensDuTri spécifie l'ordre du tri : croissant ou décroissant. Si sensDuTri est égal au symbole “supérieur à” (>), l'ordre du tri est croissant. Si sensDuTri est égal au symbole “inférieur à” (<), l'ordre du tri est décroissant. Si sensDuTri est omis, l'ordre du tri est croissant.

Si plus d'un tableau est spécifié, les tableaux sont triés en fonction de l'ordre défini pour le premier tableau (les tris multi-niveaux ne sont pas possibles dans ce cas). Ce fonctionnement est particulièrement utile avec les zones de défilement dans des formulaires : TRIER TABLEAU assure la synchonisation des tableaux utilisés pour ces zones.

Exemples

(1) L'exemple suivant crée deux tableaux et les trie en fonction du nom de la société :

```
TOUT SELECTIONNER ([Personnes])
SELECTION VERS TABLEAU ([Personnes]Noms;tabNoms;[Personnes]Sociétés;
                                                    tabSociétés)
⇒ TRIER TABLEAU (tabSociétés; tabNoms; >)
```

Cependant, comme TRIER TABLEAU n'effectue pas de tris multi-niveaux, les noms des personnes apparaîtront en désordre à l'intérieur de chaque société. Pour que les noms des personnes soient triés pour chaque société, vous devrez plutôt écrire :

```
TOUT SELECTIONNER ([Personnes])
TRIER ([Personnes];[Personnes]Sociétés; >;[Personnes]Noms;>)
SELECTION VERS TABLEAU ([Personnes]Noms;tabNoms;[Personnes]Sociétés;
                                                    tabSociétés)
```

(2) Vous affichez les noms d'une table [Personnes] dans une fenêtre flottante. Cette liste de noms peut être triée de A vers Z ou de Z vers A en fonction du bouton sur lequel vous cliquez, dans la fenêtre. Comme il se peut que certaines personnes portent le même nom, vous avez également créé un champ [Personnes]Numéro ID qui est un champ indexé unique. Lorsque vous cliquez sur un nom dans la liste, vous voulez récupérer l'enregistrement correspondant. En utilisant un tableau synchronisé et caché des numéros d'ID, vous êtes certain d'accéder à l'enregistrement correspondant au nom sélectionné :

```
   ` Méthode objet du tableau tabNoms
Au cas ou
   : (Evenement formulaire=Sur chargement)
      TOUT SELECTIONNER([Personnes])
      SELECTION VERS TABLEAU([Personnes]Noms;tabNoms;
                                    [Personnes]Numéro ID;tabIDs)
⇒     TRIER TABLEAU(tabNoms;tabIDs;>)
   : (Evenement formulaire=Sur libération)
      EFFACER VARIABLE(tabNoms)
      EFFACER VARIABLE(tabIDs)
   : (Evenement formulaire=Sur clic souris)
      Si (tabNoms#0)
            ` Utiliser le tableau tabIDs pour récupérer le bon enregistrement
         CHERCHER([Personnes];[Personnes]Numéro IDr=tabIDs{tabNoms})
            ` Traiter ici l'enregistrement
      Fin de si
Fin de cas

   ` Méthode objet du bouton bAversZ
   ` Tri croissant des tableaux en conservant la synchronisation
⇒  TRIER TABLEAU(tabNoms;tabIDs;>)

   ` Méthode objet du bouton bZversA
   ` Tri décroissant des tableaux en conservant la synchronisation
⇒  TRIER TABLEAU(tabNoms;tabIDs;<)
```

Référence

SELECTION VERS TABLEAU, TRIER.

Chercher dans tableau

version 5

Chercher dans tableau (tableau; valeur{; début}) → Numérique

Paramètre	Type		Description
tableau	Tableau	→	Tableau dans lequel effectuer la recherche
valeur	Expression	→	Valeur de même type à rechercher dans le tableau
début	Numérique	→	Elément à partir duquel commencer la recherche
Résultat	Numérique	←	Numéro du premier élément trouvé correspondant à valeur

Description

Chercher dans tableau retourne le numéro du premier élément de tableau qui correspond à valeur. Chercher dans tableau peut être utilisé avec des tableaux de type Texte, Alpha, Numérique, Date, Pointeur et Booléen. Les paramètres tableau et valeur doivent être du même type.

Si aucun élément n'est trouvé, Chercher dans tableau renvoie –1.

Si début est spécifié, Chercher dans tableau commence la recherche à l'élément spécifié par début. Si début n'est pas spécifié, Chercher dans tableau commence la recherche à l'élément 1.

Exemples

(1) La méthode projet suivante efface tous les éléments vides du tableau alpha ou texte passé en paramètre :

```
      ` Méthode projet NETTOYER TABLEAU
      ` NETTOYER TABLEAU ( Pointeur )
      ` NETTOYER TABLEAU ( -> Tableau Texte ou Alpha )

   C_POINTEUR ($1)
   Repeter
⇒     $vlElem:=Chercher dans tableau ($1->;"")
      Si ($vlElem>0)
         SUPPRIMER LIGNES ($1->;$vlElem)
      Fin de si
   Jusque ($vlElem<0)
```

Une fois que cette méthode projet est implémentée dans votre base, vous pouvez écrire, par exemple :

```
TABLEAU TEXTE (TabValeurs;...)
   ` ...
   ` Utiliser le tableau comme vous voulez
   ` ...
   ` Eliminer les éléments chaînes vides
NETTOYER TABLEAU (->TabValeurs)
```

(2) La méthode projet suivante sélectionne le premier élément d'un tableau dont le pointeur passé comme premier paramètre correspond à la valeur de la variable ou du champ dont le pointeur est passé en second paramètre :

```
   ` Méthode projet SELECTIONNER ELEMENT
   ` SELECTIONNER ELEMENT ( Pointeur ; Pointeur)
   ` SELECTIONNER ELEMENT ( -> Tableau Texte ou Alpha ; -> Champ ou variable
                                                          de type Texte ou Alpha )
```

```
⇒  $1->:=Chercher dans tableau ($1->;$2->)
   ` Si aucun élément n'est trouvé, fixer le tableau à aucun élément sélectionné
   Si ($1->=-1)
      $1->:=0
   Fin de si
```

Une fois que cette méthode projet est implémentée dans la base, vous pouvez écrire, par exemple :

```
   ` Méthode objet du pop-up menu TabTitres
   Au cas ou
      : (Evenement formulaire=Sur chargement)
         SELECTIONNER ELEMENT (->TabTitres;->[Personnes]Titre)
   Fin de cas
```

Référence

INSERER LIGNES, SUPPRIMER LIGNES, Taille tableau.

INSERER LIGNES (tableau; position{; combien})

Paramètre	Type		Description
tableau	Tableau	→	Nom du tableau dans lequel insérer des éléments
position	Numérique	→	Position de départ du ou des élément(s) à insérer
combien	Numérique	→	Nombre d'éléments à insérer ou 1 élément si ce paramètre est omis

Description

INSERER LIGNES insère un ou plusieurs éléments ou "lignes" dans le tableau tableau. Les nouveaux éléments sont insérés avant l'élément spécifié par position, et initialisés à la valeur vide du type du tableau. Tous les éléments situés au-delà de position sont décalés vers le bas d'un offset ou de la valeur spécifiée par combien.

Si position est supérieur à la taille du tableau, les éléments sont insérés à la fin du tableau.

Le paramètre combien représente le nombre de lignes à insérer. Si combien n'est pas spécifié, un seul élément est inséré. La taille du tableau est augmentée de combien.

Exemples

(1) L'exemple suivant insère cinq nouveaux éléments à partir de l'élément 10 :

```
⇒ INSERER LIGNES (unTableau; 10; 5)
```

(2) L'exemple suivant ajoute un élément à un tableau :

```
   $vlElem:=Taille tableau (unTableau)+1
⇒  INSERER LIGNES (unTableau;$vlElem)
   unTableau{$vlElem}:=...
```

Référence

SUPPRIMER LIGNES, Taille tableau.

SUPPRIMER LIGNES (tableau; position{; combien})

Paramètre	Type		Description
tableau	Tableau	→	Tableau dans lequel supprimer des lignes
position	Numérique	→	Elément de départ de la suppression
combien	Numérique	→	Nombre d'éléments à supprimer ou 1 élément si ce paramètre est omis

Description

La commande SUPPRIMER LIGNES supprime un ou plusieurs élément(s) de tableau. Les éléments sont supprimés à partir de l'élément spécifié par position.

Le paramètre combien est le nombre d'éléments à supprimer. Si combien n'est pas spécifié, un seul élément est supprimé. La taille du tableau est réduite de combien.

Exemples

(1) L'exemple suivant supprime trois éléments en commençant à l'élément 5 :

```
⇒ SUPPRIMER LIGNES (unTableau; 5; 3)
```

(2) L'exemple suivant supprime le dernier élément d'un tableau, s'il existe :

```
   $vlElem:=Taille tableau (unTableau)
   Si ($vlElem>0)
⇒     SUPPRIMER LIGNES (unTableau;$vlElem)
   Fin de si
```

Référence

INSERER LIGNES, Taille tableau.

COPIER TABLEAU (source; destination)

Paramètre	Type		Description
source	Tableau	→	Tableau à recopier
destination	Tableau	←	Tableau de destination

Description

La commande COPIER TABLEAU crée ou remplace le tableau destination avec les mêmes contenu, taille et type que le tableau source.

Les tableaux source et destination peuvent être des tableaux locaux, process ou interprocess. La portée du tableau n'a pas d'importance lors de la duplication des tableaux.

Exemple

L'exemple suivant remplit un tableau C. Un nouveau tableau, "D", est ensuite créé, contenant les mêmes informations que le tableau C :

```
   ` Sélectionner tous les enregistrements dans la table
TOUT SELECTIONNER ([Contacts])
   ` Remplir le tableau C avec les données du champ
SELECTION VERS TABLEAU ([Contacts]Société; C)
⇒  COPIER TABLEAU (C; D)  ` Copier le tableau C dans le tableau D
```

ENUMERATION VERS TABLEAU

Tableaux

version 5

Note de compatibilité
En raison de la nouvelle implémentation des listes hiérarchiques, la compatibilité de cette fonction n'a pu être totalement maintenue. Aussi, à compter de la version 6, il est préférable d'utiliser la fonction Charger liste pour travailler avec des listes hiérarchiques définies dans l'éditeur d'énumérations, en mode Structure.

ENUMERATION VERS TABLEAU (énumération; tableau{; réfEléments})

Paramètre	Type		Description
énumération	Alpha	→	Enumération de laquelle copier les éléments du premier niveau
tableau	Tableau	←	Tableau dans lequel copier les éléments de l'énumération
réfEléments	Tableau Num	←	Numéros de référence des éléments de l'énumération

Description

La commande ENUMERATION VERS TABLEAU crée ou remplace le tableau tableau avec les éléments du premier niveau de l'énumération énumération.

Si vous n'avez pas préalablement défini le tableau comme tableau de type Alpha ou Texte, ENUMERATION VERS TABLEAU crée un tableau de type Texte par défaut.

Le paramètre optionnel réfEléments (un tableau de type Numérique) retourne les numéros de référence des éléments de l'énumération.

Note de compatibilité : Dans la version précédente de 4D, le troisième paramètre était un tableau rempli avec les noms de toutes les énumérations liées. Si un élément de l'énumération principale comportait une énumération liée, le nom de cette dernière était placée dans l'élément de tableau ayant le même numéro que l'élément de l'énumération. S'il n'y avait pas d'énumération liée, l'élément inséré était une chaîne vide. Ce second tableau avait la même taille que tableau. Vous pouviez utiliser les noms dans le tableau liens pour accéder aux énumérations liées.

Vous pouvez continuer à utiliser ENUMERATION VERS TABLEAU pour construire un tableau basé sur les éléments de premier niveau d'une énumération. Cependant, cette commande ne vous donne pas les moyens de travailler avec les éléments des sous-énumérations. Pour exploiter pleinement les listes hiérarchiques, utilisez les nouvelles commandes de listes hiérarchiques introduites par la version 6.

Exemple

L'exemple suivant recopie les éléments de l'énumération Régions dans le tableau tabRégions :

```
⇒ ENUMERATION VERS TABLEAU ("Régions"; tabRégions)
```

Référence

Charger liste, STOCKER LISTE, TABLEAU VERS ENUMERATION.

TABLEAU VERS ENUMERATION

version 5

Note de compatibilité
En raison de la nouvelle implémentation des listes hiérarchiques, la compatibilité de cette commande n'a pu être totalement maintenue. Aussi, à compter de la version 6, il est préférable d'utiliser la commande STOCKER LISTE pour travailler avec des listes hiérarchiques définies dans l'éditeur d'énumérations, en mode Structure.

TABLEAU VERS ENUMERATION (tableau; énumération{; réfEléments})

Paramètre	Type		Description
tableau	Tableau	→	Tableau duquel copier les éléments
énumération	Alpha	←	Enumération dans laquelle copier les éléments de tableau
réfEléments	Tableau Num	→	Tableau numérique des numéros de référence des éléments

Description

La commande TABLEAU VERS ENUMERATION crée ou remplace l'énumération énumération (définie dans l'éditeur d'énumérations en mode Structure) en utilisant les éléments du tableau tableau.

Cette commande vous permet de définir seulement les éléments du premier niveau de l'énumération.

Le paramètre optionnel réfEléments, s'il est passé, doit être un tableau de type Numérique synchronisé avec le tableau tableau. Chaque élément de ce tableau indique le numéro de référence de l'élément de l'énumération correspondant dans tableau. Si ce paramètre est omis, 4D affecte automatiquement aux éléments de l'énumération les numéros de référence 1, 2... N.

Note de compatibilité : Dans la version précédente de 4D, ce paramètre était utilisé pour lier d'autres énumérations à chaque élément de tableau. Si un élément du tableau liens correspondait au nom d'une énumération existante, cette énumération était alors rattachée à l'élément correspondant.

Vous pouvez continuer à utiliser TABLEAU VERS ENUMERATION pour construire une énumération basée sur les éléments d'un tableau. Cependant, cette commande ne vous donne pas les moyens de travailler avec les éléments des sous-énumérations. Pour exploiter pleinement les listes hiérarchiques, utilisez les nouvelles commandes de listes hiérarchiques introduites avec la version 6 de 4D.

Exemple

L'exemple suivant copie le tableau tabRégions dans l'énumération "Régions" :

⇒ **TABLEAU VERS ENUMERATION** (tabRégions; "Régions")

Référence

APPELER SUR ERREUR, Charger liste, ENUMERATION VERS TABLEAU, STOCKER LISTE.

Gestion des erreurs

La commande TABLEAU VERS ENUMERATION génère l'erreur -9957 lorsqu'elle est appliquée à une énumération en cours de modification en mode Structure. Vous pouvez intercepter cette erreur à l'aide d'une méthode projet de gestion des erreurs installée par la commande APPELER SUR ERREUR.

SELECTION VERS TABLEAU

SELECTION VERS TABLEAU (champ | table; tableau{; champ2 | table2; tableau2; ...; champN | tableN; tableauN})

Paramètre	Type		Description
champ | table	Champ | Table	→	Champ à récupérer dans le tableau ou Table dont les numéros d'enregistrements sont à récupérer dans le tableau
tableau	Tableau	←	Tableau recevant les valeurs des champs ou les numéros d'enregistrements

Description

La commande SELECTION VERS TABLEAU crée un ou plusieurs tableaux et y copie les valeurs de champ(s) ou les numéros d'enregistrement(s) de la sélection courante.

SELECTION VERS TABLEAU s'applique à la sélection courante de la table spécifiée dans le premier paramètre. La commande peut réaliser les opérations suivantes :
• Charger les valeurs d'un ou plusieurs champs,
• Charger les numéros des enregistrements, à l'aide de la syntaxe ...;[table];tableau;...
• Charger des valeurs de champs liés, si il existe un lien automatique de N vers 1 entre les tables, ou si vous avez préalablement appelé la commande LIENS AUTOMATIQUES pour rendre automatiques les liens manuels N vers 1 (dans les deux cas, les valeurs peuvent être chargées à travers plusieurs niveaux de liens N vers 1 entre les tables).

Chaque tableau est typé en fonction du type de champ. Il y a cependant deux exceptions :
• Lorsqu'un champ de type Texte est copié dans un tableau Alpha, le tableau reste de type Alpha.
• La copie d'un champ de type Heure provoquera la création d'un tableau Entier long.

Note : Vous ne pouvez pas utiliser de champs de type Sous-table ni de sous-champs.

Si vous chargez les numéros des enregistrements, ils sont copiés dans un tableau de type Entier long.

4D Server : La commande SELECTION VERS TABLEAU est optimisée pour 4D Server. Chaque tableau est créé sur le serveur puis envoyé en totalité sur le poste client.

ATTENTION : SELECTION VERS TABLEAU peut créer des tableaux de taille importante, en fonction de la taille de la sélection courante, ainsi que du type et de la taille des données à charger. Comme les tableaux résident en mémoire, il peut être utile de tester la taille des tableaux créés après l'exécution de la commande, ou d'utiliser une méthode projet d'appel sur erreur.

Note : Après un appel à SELECTION VERS TABLEAU, la sélection courante et l'enregistrement courant ne sont pas modifiés, mais l'enregistrement courant n'est plus chargé. Utilisez la commande CHARGER ENREGISTREMENT après un SELECTION VERS TABLEAU si vous souhaitez utiliser les valeurs des champs de l'enregistrement courant.

Exemples

(1) Dans l'exemple suivant, la table [Personnes] dispose d'un lien automatique vers la table [Sociétés]. Les deux tableaux tabNoms et tabAdresseSociétés sont dimensionnés en fonction du nombre d'enregistrements dans la sélection de la table [Personnes] et contiennent des informations venant des deux tables :

```
⇒ SELECTION VERS TABLEAU ([Personnes]Nom; tabNoms; [Sociétés]Adresse;
                                                    tabAdresseSociétés)
```

(2) L'exemple ci-dessous retourne les numéros d'enregistrements de la table [Clients] dans le tableau tabNumEnr et les valeurs du champ [Clients]Noms dans le tableau tabNoms :

```
⇒ SELECTION VERS TABLEAU([Clients]; tabNumEnr; [Clients]Noms; tabNoms)
```

Référence

APPELER SUR ERREUR, LIENS AUTOMATIQUES, SOUS SELECTION VERS TABLEAU, TABLEAU VERS SELECTION.

SELECTION LIMITEE VERS TABLEAU (début; fin; champ | table; tableau{; champ2 | table2; tableau2; ...; champN | tableN; tableauN})

Paramètre	Type		Description
début	Numérique	→	Numéro de l'enregistrement sous-sélectionné à partir duquel commencer la copie des données
fin	Numérique	→	Numéro de l'enregistrement sous-sélectionné auquel arrêter la copie des données
champ | table	Champ | Table	→	Champ à utiliser pour récupérer les données ou Table à utiliser pour récupérer les numéros d'enregistrements
tableau	Tableau	←	Tableau recevant les données ou les numéros d'enregistrements

Description

SELECTION LIMITEE VERS TABLEAU crée un ou plusieurs tableaux et y copie des données en provenance des champs de la sélection courante ou les numéros des enregistrements de la sélection courante.

A la différence de SELECTION VERS TABLEAU qui s'applique à l'intégralité de la sélection courante, SELECTION LIMITEE VERS TABLEAU s'applique uniquement à une sous-sélection d'enregistrements, définie par les paramètres début et fin.

Vous devez passer dans les paramètres début et fin des numéros d'enregistrements sous-sélectionnés s'inscrivant dans l'intervalle défini par la formule 1 <= début <= fin <= Enregistrements trouves ([...]).

Si vous passez des numéros correspondant à 1 <= début = fin <= Enregistrements trouves ([...]), ce sont les champs ou les numéros des enregistrements de la sélection courante répondant à début = fin qui seront chargés.

Si vous passez des numéros d'enregistrements incorrects, vous obtiendrez les résultats suivants :
• Si fin > Enregistrements trouves ([...]), la commande retourne toutes les valeurs, à partir l'enregistrement sous-sélectionné spécifié par début jusqu'au dernier enregistrement sous-sélectionné.
• Si début > fin, la commande ne retourne que les valeurs de l'enregistrement début.

• Si les deux paramètres incompatibles avec la taille de la sous-sélection, les tableaux sont retournés vides

Comme SELECTION VERS TABLEAU, SELECTION LIMITEE VERS TABLEAU s'applique à la sélection de la table passée en paramètre. La commande peut réaliser les opérations suivantes :
• Charger les valeurs d'un ou plusieurs champs,
• Charger les numéros des enregistrements, à l'aide de la syntaxe ...;[table];tableau;...
• Charger des valeurs de champs liés, si il existe un lien automatique de N vers 1 entre les tables, ou si vous avez préalablement appelé la commande LIENS AUTOMATIQUES pour rendre automatiques les liens manuels N vers 1 (dans les deux cas, les valeurs peuvent être chargées à travers plusieurs niveaux de liens N vers 1 entre les tables).

Chaque tableau est typé en fonction du type de champ. Il y a cependant deux exceptions :
• Lorsqu'un champ de type Texte est copié dans un tableau Alpha, le tableau reste de type Alpha.
• La copie d'un champ de type Heure provoquera la création d'un tableau Entier long.

Note : Vous ne pouvez pas utiliser de champs de type Sous-table ni de sous-champs.

Si vous chargez les numéros des enregistrements, ils sont copiés dans un tableau de type Entier long.

4D Server : La commande SELECTION LIMITEE VERS TABLEAU est optimisée pour 4D Server. Chaque tableau est créé sur le serveur puis envoyé en totalité sur le poste client.

ATTENTION : SELECTION LIMITEE VERS TABLEAU peut créer des tableaux de taille importante, en fonction de l'intervalle défini par début et fin, ainsi que du type et de la taille des données à charger. Comme les tableaux résident en mémoire, il peut être utile de tester la taille des tableaux créés après l'exécution de la commande, ou d'utiliser une méthode projet d'interception d'erreurs, installée par la commande APPELER SUR ERREUR.

Une fois la commande correctement exécutée, la taille des tableaux résultants est égale à (fin-début)+1 — sauf si le paramètre fin est supérieur au nombre d'enregistrements dans la sélection. Dans ce cas, les tableaux contiennent (Enregistrements trouves([...])-début)+1 éléments.

Exemples

(1) La ligne de code suivante utilise les 50 premiers enregistrements de la sélection courante de la table [Factures]. Les valeurs du champ [Factures]RéfFacture et du champ lié [Clients]RéfClient sont chargées.

```
⇒ SELECTION LIMITEE VERS TABLEAU (1;50;[Factures]RéfFacture;tlRéfFacture;
                                  [Clients]RéfClient;tlRéfClient)
```

(2) Les lignes de code suivantes utilisent les 50 derniers enregistrements de la sélection courante de la table [Factures]. Les numéros d'enregistrements de la table [Factures] ainsi que ceux de la table liée [Clients] sont chargés :

```
    lTailleSél := Enregistrements trouves ([Factures])
⇒   SELECTION LIMITEE VERS TABLEAU (lTailleSél-49;lTailleSél;[Factures];taFactureNum;
                                                                [Clients];taClientNum)
```

(3) Les lignes de code suivantes vous permettent de travailler séquentiellement avec des portions de 1000 enregistrements d'une sélection importante qui ne peut pas être chargée dans des tableaux en une seule fois :

```
    lMaxPage := 1000
    lTailleSél := Enregistrements trouves ([Annuaire])
    Boucle ($lPage ; 1; 1+((lTailleSél-1)\lMaxPage) )
        ` Charger les valeurs et/ou les numéros d'enregistrements
⇒       SELECTION LIMITEE VERS TABLEAU (1+(lMaxPage*($lPage-1));
                                                lMaxPage*$lPage;...;...;...;...;...;...)
        ` Faire quelque chose avec les tableaux
    Fin de boucle
```

Référence

APPELER SUR ERREUR, LIENS AUTOMATIQUES, SELECTION VERS TABLEAU.

TABLEAU VERS SELECTION

TABLEAU VERS SELECTION (tableau; champ{; tableau2; champ2; ...; tableauN; champN})

Paramètre	Type		Description
tableau	Tableau	→	Tableau à copier dans la sélection
champ	Champ	←	Champ recevant les valeurs du tableau

Description

La commande TABLEAU VERS SELECTION copie un ou plusieurs tableaux vers une sélection d'enregistrements. Tous les champs listés doivent appartenir à la même table.

Si une sélection existe au moment de l'appel, les éléments du tableau sont copiés dans les enregistrements en fonction de l'ordre du tableau et l'ordre des enregistrements. Si le nombre d'éléments du tableau est supérieur au nombre d'enregistrements de la sélection courante, de nouveaux enregistrements sont créés. Les enregistrements, qu'ils soient nouveaux ou existants, sont automatiquement sauvegardés.

Si les tableaux ont des tailles différentes, le premier tableau détermine le nombre d'éléments à copier. Les tableaux suivants sont copiés dans les champs passés après chaque nom de tableau.

Cette commande effectue l'opération inverse de SELECTION VERS TABLEAU. Cependant, TABLEAU VERS SELECTION ne permet pas d'utiliser de champs en provenance de tables différentes ni de tables liées, même si un lien automatique existe.

ATTENTION : Comme TABLEAU VERS SELECTION remplace les informations éventuellement présentes dans les enregistrements existants, cette commande doit être utilisée avec prudence. Si un enregistrement est verrouillé par un autre process pendant l'exécution de la commande TABLEAU VERS SELECTION, il n'est pas modifié. Tous les enregistrements verrouillés sont placés dans l'ensemble LockedSet. Après l'exécution de TABLEAU VERS SELECTION, vous pouvez tester si l'ensemble LockedSet contient des enregistrements qui étaient verrouillés.

4D Server : Cette commande est optimisée pour 4D Server. Le tableau est envoyé au serveur depuis le poste client. Les enregistrements sont modifiés ou créés sur le serveur. Comme une telle requête est gérée de façon synchrone, le poste client doit attendre que l'opération se soit correctement déroulée. Dans les environnements multi-utilisateurs et multi-process, aucun enregistrement verrouillé ne sera réécrit.

Exemple

Dans l'exemple suivant, les deux tableaux tabNoms et tabSociétés écrivent des données dans la table [Personnes]. Les valeurs du tableau tabNoms sont placées dans le champ [Personnes]Nom et les valeurs du tableau tabSociétés sont placées dans le champ [Personnes]Société :

```
⇒ TABLEAU VERS SELECTION (tabNoms; [Personnes]Nom; tabSociétés;
                                                  [Personnes]Société)
```

Référence

SELECTION VERS TABLEAU.

VALEURS DISTINCTES

Tableaux

version 6.5 (Modifiée)

VALEURS DISTINCTES (champ; tableau)

Paramètre	Type		Description
champ	Champ | Sous-champ	→	Champ ou sous-champ à utiliser
tableau	Tableau	←	Tableau devant recevoir les données du champ indexable

Description

VALEURS DISTINCTES crée et remplit le tableau tableau avec toutes les valeurs distinctes provenant du champ champ pour la sélection courante de la table à laquelle le champ ou sous-champ appartient.

Vous pouvez passer à cette commande tout type de champ **indexable**, c'est-à-dire dont le type supporte l'indexation mais qui n'est pas forcément indexé. Toutefois, l'exécution de la commande avec des champs non indexés est plus lente qu'avec des champs indexés. A noter également que dans ce cas, la commande perd l'enregistrement courant.
Vous pouvez également exécuter cette commande avec un sous-champ. Dans ce cas, le sous-champ doit être **indexé**.

Note : La commande FIXER PARAMETRE BASE permet de définir si VALEURS DISTINCTES doit utiliser ou non l'index, en fonction nombre d'enregistrements présents dans la sélection.

Si vous passez un champ, VALEURS DISTINCTES analyse et extrait les valeurs distinctes pour les enregistrements sélectionnés uniquement. En revanche, si vous passez un sous-champ, VALEURS DISTINCTES examine tous les sous-enregistrements de chaque enregistrement sélectionné.

Note : Lorsque vous exécutez VALEURS DISTINCTES au sein d'une transaction non encore terminée, la commande tient compte des enregistrements créés au cours de la transaction.

Le tableau utilisé par VALEURS DISTINCTES doit être du même type que le champ ou sous-champ passé en premier paramètre, sinon le tableau est retypé. Il y a une exception à cette règle : si le champ est de type Heure, le tableau correspondant doit être de type Entier long.

Après l'appel, la taille du tableau est égale au nombre de valeurs distinctes trouvées dans la sélection. La commande ne modifie pas la sélection courante ni l'enregistrement courant. Les éléments dans tableau sont triés par ordre croissant car VALEURS DISTINCTES utilise l'index du champ. Si cet ordre vous convient, vous n'avez donc pas besoin d'appeler TRIER TABLEAU après l'exécution de VALEURS DISTINCTES.

ATTENTION : VALEURS DISTINCTES peut créer des tableaux de taille importante, en fonction de la taille de la sélection courante, ainsi que du type et de la taille des données à charger. Comme les tableaux résident en mémoire, il peut être utile de tester la taille des tableaux créés après l'exécution de la commande, ou d'utiliser une méthode projet d'interception d'erreurs installée par la commande APPELER SUR ERREUR.

4D Server : Cette commande est optimisée pour 4D Server. Le tableau est créé et les valeurs sont calculées sur le serveur. Seul le tableau est envoyé au client.

Exemples

(1) L'exemple suivant crée une liste de villes à partir de la sélection courante et indique à l'utilisateur le nombre de villes dans lesquelles la société dispose de magasins :

```
      TOUT SELECTIONNER([Revendeurs]) ` Créer une sélection d'enregistrements
⇒     VALEURS DISTINCTES([Revendeurs]Ville;taVilles)
      ALERTE("Cette société dispose de magasins dans " +Chaine(Taille tableau(taVilles))
                                                              +" villes.")
```

(2) L'exemple suivant retourne dans taMotsClés tous les mots-clés qui sont liés (à l'aide d'une sous-table) aux documents 4D Write stockés dans la table [Documentation] et dont le sujet est “Economie” :

```
      CHERCHER ([Documentation];[Documentation]Sujet="Economie")
⇒     VALEURS DISTINCTES([Documentation]MotsClés'Mot;taMotsClés)
```

Une fois ce tableau construit, vous pouvez le réutiliser pour chercher rapidement les documents liés au mot-clé sélectionné en écrivant :

```
CHERCHER([Documentation];[Documentation]MotsClés'Mot=taMotsClés{taMotsClés})
SELECTION VERS TABLEAU ([Documentation]Sujet;taSujets)
   ` ...
```

Référence

APPELER SUR ERREUR, FIXER PARAMETRE BASE, SELECTION LIMITEE VERS TABLEAU, SELECTION VERS TABLEAU.

TABLEAU ENTIER LONG SUR SELECTION

Tableaux

version 6.7 (Modifiée)

TABLEAU ENTIER LONG SUR SELECTION (table; tabEnrg{; tempo})

Paramètre	Type		Description
table	Table	→	Table de la sélection courante
tabEnrg	Tab Entier long	←	Tableau de numéros d'enregistrements
tempo	Alpha	→	Nom de la sélection temporaire ou Sélection courante si ce paramètre est omis

Description

La commande TABLEAU ENTIER LONG SUR SELECTION remplit le tableau tabEnrg avec les numéros (absolus) des enregistrements faisant partie de la sélection temporaire tempo.

Si vous ne passez pas le paramètre tempo, la commande utilise la sélection courante de la table table.

Note : L'élément n° 0 du tableau tabEnrg est initialisé à -1.

Référence

CREER SELECTION SUR TABLEAU.

TABLEAU BOOLEEN SUR ENSEMBLE (tabBooléen{; ensemble})

Paramètre	Type		Description
tabBooléen	Tab booléen	←	Tableau d'appartenance des enregistrements à l'ensemble
ensemble	Alpha	→	Nom de l'ensemble ou Ensemble UserSet si ce paramètre est omis

Description

La commande TABLEAU BOOLEEN SUR ENSEMBLE remplit un tableau de booléens indiquant si chaque enregistrement de la table à laquelle appartient ensemble fait ou non partie de l'ensemble.

Les éléments du tableau sont ordonnés en fonction de l'ordre de création des enregistrements dans la table (numéros absolus). Si N est le nombre d'enregistrements de la table, l'élément 0 du tableau correspond à l'enregistrement n° 0, l'élément 1 du tableau correspond à l'enregistrement n° 1, etc.

Chaque élément du tableau est mis à :

- Vrai si l'enregistrement correspondant fait partie de l'ensemble,
- Faux si l'enregistrement correspondant ne pas partie de l'ensemble.

Attention, le total nombre d'éléments du tableau tabBooléen n'est pas significatif. En effet, pour des raisons structurelles, il peut être différent du nombre d'enregistrements effectivement présents dans la table. Les éventuels éléments supplémentaires sont mis à Faux.

Si vous ne passez pas le paramètre ensemble, la commande utilisera l'ensemble système *UserSet* du process courant.

Référence

CREER ENSEMBLE SUR TABLEAU.

53

Transactions

Les transactions sont une série de modifications effectuées à l'intérieur d'un process sur des données reliées entre elles. Une transaction n'est sauvegardée de façon définitive dans la base que si la transaction est validée. Si une transaction n'est pas complétée, parce qu'elle est annulée ou en raison d'un quelconque événement extérieur, les modifications ne sont pas sauvegardées.

Pendant une transaction, toutes les modifications effectuées sur les données de la base dans le process sont stockées localement dans un buffer temporaire. Si la transaction est acceptée avec VALIDER TRANSACTION, les changements sont sauvegardés de façon définitive. Si la transaction est annulée avec ANNULER TRANSACTION, les changements ne sont pas sauvegardés.

Après la validation ou l'annulation d'une transaction, la sélection de chaque table pour le process courant devient vide car les transactions manipulent des adresses d'enregistrements temporaires. Pour la même raison, soyez vigilants en utilisant des sélections temporaires au cours d'une transaction. Après validation ou annulation d'une transaction, une sélection temporaire créée avant ou pendant la transaction peut comporter des adresses d'enregistrements incorrectes. Par exemple, une sélection temporaire peut conserver les adresses d'enregistrements détruits ou l'adresse temporaire d'un enregistrement créé pendant la transaction. Cela s'applique aussi aux ensembles puisqu'ils sont basés sur des tables de bits avec des adresses d'enregistrements.

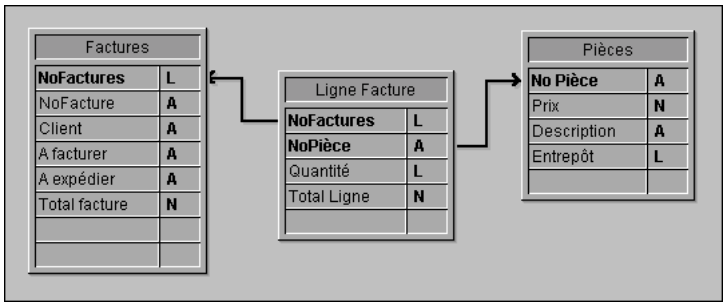
Note : Les enregistrements créés lors d'une transaction reçoivent des numéros temporaires qui s'incrémentent à partir du numéro 18 000 000.

Les commandes suivantes utilisent des numéros d'enregistrements. Leur utilisation dans une transaction est donc déconseillée :
- ALLER A ENREGISTREMENT
- JOINTURE
- SELECTION RETOUR
- REDUIRE SELECTION
- SCAN INDEX

Exemples de transactions

L'exemple de cette section s'appuie sur la structure présentée ci-dessous. C'est une base relativement simple de facturation. Les lignes de factures sont stockées dans une table appelée [Ligne Facture], qui est reliée à la table [Factures] par une relation entre les champs [Factures]NoFacture et [Ligne Facture]NoFacture. Lorsqu'une facture est ajoutée, un numéro unique est calculé avec la commande Numerotation automatique. Le lien entre [Factures] et [Ligne Facture] est du type aller-retour automatique. L'option "Mise à jour auto dans les sous-formulaires" est cochée.

Le lien entre [Ligne Facture] et [Pièces] est manuel.



Quand un utilisateur saisit une facture, les actions suivantes doivent être exécutées:
• Ajouter un enregistrement dans la table [Factures].
• Ajouter plusieurs enregistrements dans la table [Ligne Facture].
• Mettre à jour le champ [Pièces]Entrepôt pour chaque pièce figurant sur la facture.

En d'autres termes, vous devez sauvegarder les données liées. C'est la situation type où vous devez utiliser une transaction. Vous pourrez ainsi être certain de pouvoir sauvegarder tous ces enregistrements pendant l'opération ou d'avoir la possibilité d'annuler la transaction si un enregistrement ne peut être ajouté ou mis à jour.

Si vous n'utilisez pas une transaction, vous ne pouvez pas garantir l'intégrité logique des données de votre base. Par exemple, si un enregistrement parmi ceux de la table [Pièces] est verrouillé, vous ne pourrez pas mettre à jour la quantité stockée dans le champ [Pièces]Entrepôt. Ce champ sera alors logiquement incorrect. La somme des pièces vendues et restantes dans l'entrepôt ne sera pas égale à la quantité d'orgine saisie dans l'enregistrement. Vous pouvez éviter cette situation en utilisant les transactions.

Il y a plusieurs façons d'effectuer une saisie sous transaction :

(1) Vous pouvez laisser 4D gérer les transactions à votre place en sélectionnant l'option **Transaction automatique en saisie** dans la boîte de dialogue des propriétés de la base en mode Structure. Dans ce cas, pendant la saisie, 4D ouvre si nécessaire une transaction puis la valide ou l'annule selon que vous avez accepté ou non la saisie des données. Par exemple, la saisie de données dans un formulaire contenant une table liée dans un sous-formulaire est une opération qui requiert une transaction. Cette option s'applique à toute la base.

Si vous voulez gérer les transactions vous-même, vous devez utiliser les commandes de transaction DEBUT TRANSACTION, VALIDER TRANSACTION et ANNULER TRANSACTION.

(2) Vous pouvez par exemple écrire :

```
LECTURE ECRITURE([Ligne Facture])
LECTURE ECRITURE([Pièces])
FORMULAIRE ENTREE([Factures];"Saisie")
Repeter
   DEBUT TRANSACTION
   AJOUTER ENREGISTREMENT([Factures])
   Si (OK=1)
      VALIDER TRANSACTION
   Sinon
      ANNULER TRANSACTION
   Fin de si
Jusque (OK=0)
LECTURE SEULEMENT(*)
```

(3) Pour réduire les verrouillages des enregistrements pendant la saisie de données, vous pouvez aussi choisir de gérer les transactions à partir de la méthode du formulaire et d'accéder aux tables en LECTURE ECRITURE uniquement quand cela est nécessaire.

Vous effectuez la saisie de données en utilisant le formulaire de saisie pour [Factures], qui contient la table liée [Factures]Lignes dans un sous-formulaire. Le formulaire comporte deux boutons : bAnnuler et bOK. Aucune action ne leur est attribuée.

La boucle d'ajout devient alors :

```
LECTURE ECRITURE([Ligne Facture])
LECTURE SEULEMENT([Pièces])
FORMULAIRE ENTREE([Factures];"Input")
Repeter
   AJOUTER ENREGISTREMENT([Factures])
Jusque (bOK=0)
LECTURE SEULEMENT([Ligne Facture])
```

Notez que la table [Pièces] est désormais en "lecture seulement" pendant la saisie de données. L'accès en lecture/écriture ne s'active que si les données sont validées.

La transaction est ouverte dans la méthode du formulaire entrée de la table [Factures] :

```
Au cas ou
   : (Evenement formulaire=Sur chargement)
      DEBUT TRANSACTION
      [Factures]NoFactures:=Numerotation automatique([Factures]NoFactures)
   Sinon
      [Factures]Total facture:=Somme([Ligne Facture]Total ligne)
Fin de cas
```

Si vous cliquez sur le bouton bAnnuler, la saisie et la transaction doivent être annulées. Voici la méthode objet du bouton bAnnuler:

```
Au cas ou
   : (Evenement formulaire=Sur clic souris)
      ANNULER TRANSACTION
      NE PAS VALIDER
Fin de cas
```

Si vous cliquez sur le bouton bOK, la saisie et la transaction doivent être acceptées. Voici la méthode objet du bouton bOK :

```
Au cas ou
   : (Evenement formulaire=Sur clic souris)
      $NbLines:=Enregistrements trouves([Ligne Facture])
         ` Passer en lecture/écriture pour accéder à la table [Pièces]
      LECTURE ECRITURE([Pièces])
      DEBUT SELECTION([Ligne Facture])  ` Commencer à la première ligne
      $ValidTrans:=Vrai  ` Tout devrait marcher
      Tant que ($Line;1;$NbLines)  ` Pour chaque ligne
         CHARGER SUR LIEN([Ligne Facture]NoPiece)
         OK:=1  ` Vous voulez continuer
         Tant que (Enregistrement verrouille([Pièces]) & (OK=1))
               ` Essayer d'obtenir l'enregistrement en lecture/écriture
            CONFIRMER("La pièce "+[Ligne Facture]NoPiece+" est utilisée. Vous
                                                                  attendez ?")
            Si (OK=1)
               ENDORMIR PROCESS(Numero du process courant;60)
               CHARGER ENREGISTREMENT([Pièces])
            Fin de si
         Fin tant que
         Si (OK=1)
               ` Mettre à jour quantité dans l'entrepôt
            [Pièces]Entrepôt:=[Pièces]Entrepôt-[Ligne Facture]Quantité
            STOCKER ENREGISTREMENT([Pièces])  ` Sauvegarder l'enregistrement
         Sinon
            $Ligne:=$NbLines+1  ` Sortir de la boucle
            $ValidTrans:=Faux
         Fin de si
         ENREGISTREMENT SUIVANT([Ligne Facture])  ` Aller à la ligne suivante
      Fin tant que
      LECTURE SEULEMENT([Pièces])  ` Mettre la table en mode lecture seulement
      Si ($ValidTrans)
         STOCKER ENREGISTREMENT([Factures])  ` Sauvegarder les enregistrements
            ` Valider toutes modifications les modifications de la base
         VALIDER TRANSACTION
```

```
        Sinon
            ANNULER TRANSACTION  ` Tout annuler
        Fin de si
        NE PAS VALIDER  ` Quitter le formulaire
    Fin de cas
```

Dans le code ci-dessus, quel que soit le bouton sur lequel l'utilisateur a cliqué, nous appelons la commande NE PAS VALIDER. Le nouvel enregistrement n'est pas validé par un appel à VALIDER mais par STOCKER ENREGISTREMENT. De plus, vous remarquez que STOCKER ENREGISTREMENT est appelée juste avant la commande VALIDER TRANSACTION. Ainsi, la sauvegarde de l'enregistrement [Factures] est partie intégrante de la transaction. Appeler la commande VALIDER validerait aussi l'enregistrement mais dans ce cas, la transaction serait validée avant le stockage de la facture. Autrement dit, l'enregistrement serait sauvergardé en-dehors de la transaction.

En fonction de vos besoins, laissez 4D gérer les transactions pendant la saisie ou personnalisez votre base à votre convenance, comme dans les exemples précédents. Dans le dernier exemple, la gestion du verrouillage des enregistrements de la table [Pièces] pourrait être plus élaborée.

Référence

ANNULER TRANSACTION, DEBUT TRANSACTION, Transaction en cours, VALIDER TRANSACTION.

DEBUT TRANSACTION

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

DEBUT TRANSACTION débute une transaction dans le process courant. Toutes les modifications apportées à la base seront stockées temporairement jusqu'à ce que la transaction soit validée ou annulée.

Si vous avez plusieurs process globaux, vous pouvez avoir plusieurs transactions. Vous ne pouvez pas débuter des transactions dans d'autres transactions. Si vous débutez une transaction dans une autre, 4e Dimension ignore la seconde.

Référence

ANNULER TRANSACTION, Transaction en cours, Utiliser des transactions, VALIDER TRANSACTION.

VALIDER TRANSACTION

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

VALIDER TRANSACTION accepte la transaction ouverte par la commande DEBUT TRANSACTION dans le process courant. VALIDER TRANSACTION sauvegarde toutes les modifications apportées à la base pendant la transaction.

Référence

ANNULER TRANSACTION, DEBUT TRANSACTION, Transaction en cours, Utiliser des transactions.

ANNULER TRANSACTION

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

ANNULER TRANSACTION annule la transaction ouverte par la commande DEBUT TRANSACTION dans le process courant. ANNULER TRANSACTION laisse la base inchangée en annulant toutes les opérations éventuellement exécutées pendant la transaction.

Référence

DEBUT TRANSACTION, Transaction en cours, Utiliser des transactions, VALIDER TRANSACTION.

Transaction en cours → Booléen

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Booléen	←	VRAI si le process courant est en transaction, FAUX sinon

Description

La commande Transaction en cours retourne VRAI si le process courant est en transaction, sinon elle retourne FAUX.

Exemple

Si vous effectuez des opérations (ajout, modification ou suppression) sur de multiples enregistrements, vous pouvez rencontrer des enregistrements verrouillés. Dans ce cas, pour préserver l'intégrité des données, vous devez avoir ouvert une transaction, de manière à ce que vous puissiez faire “marche arrière” et annuler l'ensemble de l'opération depuis le début, sans que les données de la base soient modifiées.

Si vous effectuez l'opération depuis un trigger ou une sous-routine pouvant être appelé(e) dans une transaction ou hors transaction, l'utilisation de la commande Transaction en cours vous permet de vérifier que la méthode du process courant ou la méthode appelante a bien ouvert une transaction. Si ce n'est pas le cas, vous ne commencez même pas l'opération, car, en cas d'échec au cours du processus, vous ne pourriez pas revenir sur les opérations déjà effectuées.

Référence

ANNULER TRANSACTION, DEBUT TRANSACTION, Présentation des triggers, VALIDER TRANSACTION.

54

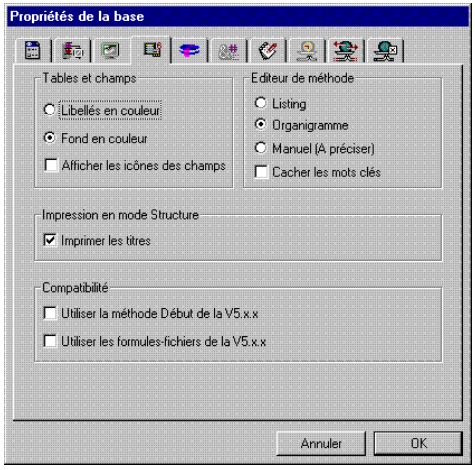
Triggers

Présentation des triggers

Un trigger est une méthode associée à une table. C'est une propriété d'une table. Vous n'appelez pas un trigger, les triggers sont appelés automatiquement par le moteur de 4D à chaque fois qu'un enregistrement de la table est manipulé (ajout, suppression, modification et chargement). Les triggers sont des méthodes qui peuvent éviter des opérations “illégales” dans votre base. Par exemple, dans une facturation, vous pouvez empêcher qu'un utilisateur crée une facture sans spécifier à qui elle doit être adressée. Les triggers sont un outil puissant permettant de contrôler les opérations sur les tables, et d'éviter des pertes de données accidentelles. Vous pouvez créer des triggers très simples et les rendre de plus en plus sophistiqués.

Compatibilité avec les versions précédentes de 4D

Les triggers sont un nouveau type de méthode introduit dans la version 6 de 4D.
Dans les versions précédentes, les méthodes table (appelées formules-fichier) étaient exécutées par 4D seulement lorsqu'un formulaire d'une table était utilisé pour la saisie de données, l'affichage ou l'impression. Elles étaient rarement utilisées. Notez que les triggers s'exécutent à un niveau beaucoup plus bas que les précédentes formules-fichier. Quelle que soit l'opération effectuée sur un enregistrement par l'intermédiaire d'une action utilisateur (comme la saisie de données) ou par programmation (comme un appel à STOCKER ENREGISTREMENT), le trigger de la table sera appelé par 4D. Les triggers sont très différents des formules-fichier. Si vous avez converti une base version 5 en version 6, et si vous voulez tirer parti des possibilités offertes par les triggers, désélectionnez l'option **Utiliser les formules-fichiers de la V5.x.x** dans la boîte de dialogue **Propriétés de la base**.



Activer et créer un trigger

Par défaut, lorsque vous créez une table en mode Structure, la table n'a pas de trigger.

Pour utiliser un trigger pour une table, vous devez :
• activer le trigger et indiquer à 4D quand l'appeler.
• créer et écrire le code pour le trigger.

Activer un trigger qui n'est pas encore écrit ou écrire un trigger sans l'activer n'affecte pas les opérations effectuées sur une table.

1. Activer un Trigger

Pour activer le trigger, sélectionnez les options **Triggers** pour la table dans la fenêtre des **Propriétés de la table** :

Voici la description de ces options :

Sur sauvegarde nouvel enreg.

Si cette option est sélectionnée, le trigger sera appelé à chaque fois qu'un enregistrement est créé dans la table, c'est-à-dire dans les circonstances suivantes :
• Vous ajoutez un enregistrement lors de la saisie de données (mode Utilisation ou commande AJOUTER ENREGISTREMENT).
• Vous créez et sauvegardez un enregistrement avec CREER ENREGISTREMENT et STOCKER ENREGISTREMENT. Notez que le trigger est appelé au moment où vous exécutez STOCKER ENREGISTREMENT, et non quand il est réellement créé.
• Vous importez des enregistrements (mode Utilisation ou commandes du langage).

• Vous appelez d'autres commandes qui créent et/ou sauvegardent de nouveaux enregistrements (par exemple, TABLEAU VERS SELECTION, STOCKER SUR LIEN, etc.).
• Vous utilisez un plug-in 4D qui appelle les commandes CREER ENREGISTREMENT et STOCKER ENREGISTREMENT.

Sur sauvegarde enregistrement
Si cette option est sélectionnée, le trigger sera appelé à chaque fois qu'un enregistrement de la table est modifié, c'est-à-dire dans les circonstances suivantes :
• Vous modifiez un enregistrement en saisie de données (mode Utilisation ou commande MODIFIER ENREGISTREMENT)
• Vous sauvegardez un enregistrement existant avec STOCKER ENREGISTREMENT.
• Vous appelez une commande qui provoque la sauvegarde d'un enregistrement existant (par exemple, TABLEAU VERS SELECTION, APPLIQUER A SELECTION, MODIFIER SELECTION, etc.)
• Vous utilisez un plug-in 4D qui appelle la commande STOCKER ENREGISTREMENT.

Sur suppression enregistrement
Si cette option est sélectionnée, le trigger sera invoqué à chaque fois qu'un enregistrement de la table est supprimé, c'est-à-dire dans les circonstances suivantes :
• Vous supprimez un enregistrement en mode Utilisation ou en appelant la commande SUPPRIMER ENREGISTREMENT ou SUPPRIMER SELECTION.
• Vous effectuez des opérations qui provoquent la suppression d'un enregistrement lié par l'intermédiaire des options de contrôle de suppression d'un lien.
• Vous utilisez un plug-in 4D qui appelle la commande SUPPRIMER ENREGISTREMENT.

Sur chargement enregistrement
Si cette option est sélectionnée, le trigger sera appelé à chaque fois qu'un enregistrement de la table est chargé. Cette option couvre toutes les situations où un enregistrement courant est chargé du fichier de données, **à l'exception des fonctions listées ci-dessous**.

Note : Ce trigger ne traite pas la création d'un nouvel enregistrement, mais uniquement le chargement d'enregistrements existants.

A des fins d'optimisation du fonctionnement de 4D, l'option Sur chargement enregistrement ne déclenche JAMAIS l'appel du trigger lors de l'utilisation des fonctions pouvant éventuellement (mais pas systématiquement) tirer parti d'un index. En effet, si l'index est utilisé, les enregistrements ne sont pas chargés. Inversement, si l'index n'est pas utilisé (par exemple si le champ traité n'est pas indexé), les enregistrements sont chargés. Cette incertitude quant à l'appel du trigger ne permet pas de l'exploiter de manière fiable.

Les fonctions pour lesquelles l'option Sur chargement enregistrement ne provoquera jamais l'appel du trigger sont les suivantes :
• Recherches : effectuées par l'utilisateur (éditeur standard) et commandes CHERCHER, CHERCHER DANS SELECTION.
• Tris : effectués par l'utilisateur (éditeur standard) et commande TRIER.
• Fonctions statistiques : Somme, Moyenne, Min, Max, Ecart type, Variance, Somme des carres.
• Commandes JOINTURE, SELECTION RETOUR.

IMPORTANT : Si vous exécutez une opération ou appelez une commande qui agit sur plusieurs enregistrements, ce trigger est appelé pour chaque enregistrement. Si, par exemple, vous appelez APPLIQUER A SELECTION pour une table dont la sélection courante est composée de 100 enregistrements, le trigger sera exécuté 100 fois.

2. Créer un trigger

Pour créer un trigger pour une table, utilisez la fenêtre de l'Explorateur ou double-cliquez sur le titre de la table dans la fenêtre de Structure en appuyant sur la touche **Alt** (sous Windows) ou **Option** (sous MacOS). Pour plus d'informations, reportez-vous au manuel *Mode Structure*.

Evénements moteur

Un trigger peut être invoqué pour l'un des quatre événements moteur décrits ci-dessus. Dans le trigger, vous détectez quel événement a lieu en appelant la fonction Evenement moteur. Cette fonction retourne une valeur numérique qui indique l'événement moteur.

Typiquement, vous écrivez un trigger avec une structure du type "Au cas ou" sur le résultat retourné par Evenement moteur :

```
   ` Trigger pour [UneTable]
C_ENTIER LONG($0)
$0:=0   ` On suppose que la requête est acceptée
Au cas ou
⇒     : (Evenement moteur=Sur sauvegarde nouvel enreg)
         ` Effectuer les actions appropriées pour sauvegarder l'enregistrement créé
⇒     : (Evenement moteur=Sur sauvegarde enregistrement)
         ` Effectuer les actions appropriées pour sauvegarder l'enregistrement déjà existant.
⇒     : (Evenement moteur=Sur suppression enregistrement)
         ` Effectuer les actions appropriées pour détruire l'enregistrement.
⇒     : (Evenement moteur=Sur chargement enregistrement)
         ` Effectuer les actions appropriées pour charger en mémoire l'enregistrement.
Fin de cas
```

Les triggers sont des fonctions

Un trigger a deux finalités :
• Effectuer des actions sur l'enregistrement juste avant qu'il soit sauvegardé, supprimé ou juste après qu'il ait été chargé.
• Accepter ou rejeter une opération de base de données.

1. Effectuer des actions

A chaque fois qu'un enregistrement est sauvegardé (ajouté ou modifié) dans une table [Documents], vous souhaitez “estampiller ” l'enregistrement avec des marqueurs de création et de modification. Vous pouvez écrire le trigger suivant :

```
   ` Trigger pour table [Documents]
Au cas ou
   : (Evenement moteur=Sur sauvegarde nouvel enreg)
      [Documents]Creation Stamp:=Time stamp
      [Documents]Modification Stamp:=Time stamp
   : (Evenement moteur=Sur sauvegarde enregistrement)
      [Documents]Modification Stamp:=Time stamp
Fin de cas
```

Note : La fonction Time stamp utilisée dans cet exemple est une petite méthode projet retournant le nombre de secondes écoulées depuis une date choisie arbitrairement.

Une fois que ce trigger a été écrit et activé, peu importe la façon dont vous ajoutez ou modifiez un enregistrement dans la table [Documents] (saisie de données, import, méthode projet, plug-in 4D), la valeur des deux champs [Documents]Creation Stamp et [Documents]Modification Stamp sera automatiquement affectée par le trigger avant que l'enregistrement ne soit écrit sur disque.

Note : Voir l'exemple de la commande PROPRIETES DOCUMENT pour une analyse complète de cet exemple.

2. Accepter ou rejeter l'opération de la base

Pour accepter ou rejeter une opération de la base, le trigger doit retourner un **code d'erreur de trigger** dans le résultat de la fonction $0.

Exemple

Prenons le cas d'une table [Employés]. Pendant la saisie de données, vous contrôlez le champ [Employés]No Séc.Soc. Par exemple, lorsque l'utilisateur clique sur le bouton de validation, vous vérifiez le champ utilisant la méthode objet du bouton :

```
   ` Méthode objet bouton bAccept
Si (Bon No Séc.Soc ([Employés]No Séc.Soc))
   VALIDER
Sinon
   BEEP
   ALERTE ("Saisissez un numéro de sécurité sociale et cliquez de nouveau sur OK.")
Fin de si
```

Si la valeur du champ est correcte, vous acceptez la saisie de données, sinon vous affichez une alerte et restez en saisie de données.

Si vous créez aussi des enregistrements pour la table [Employés] par programmation, le code ci-dessous serait valide MAIS violerait la règle imposée dans la méthode objet créée plus haut :

```
   ` Extrait d'une méthode projet
   ` ...
CREER ENREGISTREMENT ([Employés])
[Employés]Nom :="DOE"
STOCKER ENREGISTREMENT ([Employés])  ` <- violation de la règle ! Il n'y a pas de
                                     ` numéro de sécurité sociale!
```

En utilisant un trigger pour la table [Employés], vous pouvez appliquer la contrainte sur [Employés]No Séc.Soc à tous les niveaux de la base. Le trigger serait du type :

```
   ` Trigger pour [Employés]
$0:=0
$dbEvent:=Evenement moteur
Au cas ou
   : (($dbEvent=Sur sauvegarde nouvel enreg) | ($dbEvent=
                                        Sur sauvegarde enregistrement))
   Si (Non(Bon No Séc.Soc ([Employés]No Séc.Soc)))
      $0:=-15050
   Sinon
      ` ...
   Fin de si
      ` ...
Fin de cas
```

Une fois que ce trigger est écrit et activé, la ligne STOCKER ENREGISTREMENT([Employés]) de la méthode projet ci-dessus génèrera une erreur moteur -15050 et l'enregistrement ne sera PAS sauvegardé.

De la même façon, si un plug-in 4D essayait de sauvegarder un enregistrement dans [Employés] avec un numéro de sécurité sociale incorrect, le trigger génèrerait la même erreur et l'enregistrement ne serait pas sauvegardé non plus.

Le trigger garantit que personne (utilisateur, développeur, plug-in, 4D Open client avec 4D Server) ne peut violer la règle sur le numéro de sécurité sociale (à dessein ou par erreur).

Notez que même si vous n'avez pas créé de trigger pour une table, la base peut retourner des erreurs moteur lorsque vous essayez de sauvegarder ou de détruire un enregistrement. Vous pouvez, par exemple, recevoir l'erreur -9998, si vous essayez de sauvegarder un enregistrement.

Les triggers retournent de nouveaux types d'erreurs dans 4D :
• 4D gère les erreurs “normales” : index unique, contrôles relationnels, etc.
• En utilisant les triggers, vous pouvez créer des codes d'erreurs propres au contenu de votre application.

Important : Vous pouvez retourner le code d'erreur de votre choix. Cependant, n'utilisez pas des codes d'erreurs déjà utilisés par le moteur de 4D. Nous vous recommandons fortement d'utiliser des codes compris entre -32000 et -15000. Nous réservons les erreurs supérieures à –15000 au moteur de 4D.

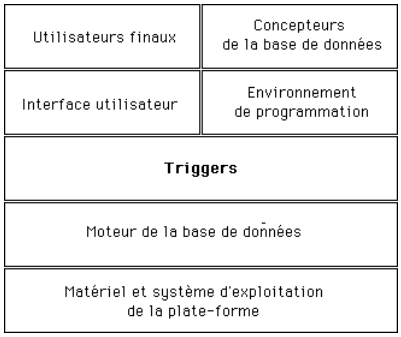
Au niveau du process, vous gérez les erreurs trigger de la même façon que les erreurs du moteur de base de données :
• vous pouvez laisser 4D afficher la boîte de dialogue standard d'erreur, la méthode est alors interrompue.
• vous pouvez utiliser une méthode de gestion d'erreur installée par APPELER SUR ERREUR et traiter l'erreur de façon appropriée.

Note : Pendant la saisie, si une erreur trigger est retournée au moment où vous essayez de valider ou de supprimer un enregistrement, l'erreur est gérée comme une erreur sur un index unique. La boîte de dialogue d'erreur est affichée et vous restez en saisie de données. Même si vous n'utilisez une base qu'en mode Utilisation (et non en Menus créés), vous bénéficiez des triggers.

Même si un trigger ne retourne pas d'erreur ($0:=0), cela ne signifie pas qu'une opération de la base s'effectuera correctement. Il peut y avoir eu un doublon sur l'index unique. Si l'opération est la mise à jour d'un enregistrement, ce dernier peut être verrouillé, une erreur d'entrée/sortie peut se produire, bien d'autres choses encore peuvent arriver. Ces vérifications sont effectuées après l'exécution du trigger. Cependant, du point de vue du plus haut niveau du process en exécution, les erreurs retournées par le moteur de la base de données ou celle d'un trigger sont de même nature : une erreur trigger est une erreur du moteur de la base de données.

Les triggers et l'architecture 4D

Les triggers fonctionnent au niveau du moteur de la base de données. Ce point est illustré dans le schéma suivant :



Les triggers sont exécutés **sur la machine où est situé le moteur de la base de données**. Si ce point est une évidence dans le cas de 4D en mono-utilisateur, il convient de rappeler que pour 4D Server, les triggers sont exécutés sur la machine serveur (dans le process actif) et non sur la machine cliente.

Quand un trigger est appelé, il s'exécute dans le contexte du process de base de données qui tente l'opération. Ci-dessous, ce process est appelé **process appelant** l'exécution du trigger.

En particulier, le trigger fonctionne avec la sélection courante, les enregistrements courants, les modes lecture/écriture, les verrouillages d'enregistrements du process appelant.

S'il est vrai que les triggers peuvent, par exemple, tester les sémaphores globaux, soyez prudent lorsque vous utilisez les autres objets de la base et du langage, car un trigger peut s'exécuter sur une machine différente que celle du process appelant : c'est le cas avec 4D Server !

• **Variables interprocess** : Un trigger a accès aux variables interprocess de la machine où il est exécuté. Avec 4D Server, ce peut être une machine différente de celle du process appelant.
• **Variables process** : une table indépendante de variables process peut être partagée par tous les triggers. Un trigger n'a pas accès aux variables process du process appelant.
• **Variables locales** : vous pouvez utiliser des variables locales dans un trigger. Leur aire d'action est l'exécution du trigger (elles sont créées/détruites au cours de cette exécution).

• **Sémaphores** : Un trigger peut tester ou placer des sémaphores globaux et locaux (sur la machine où il s'exécute dans ce dernier cas). Cependant, un trigger doit s'exécuter rapidement. En conséquence, utilisez plutôt des sémaphores locaux dans un trigger, sauf si vous avez une idée précise en tête.
• **Ensembles et sélections temporaires** : Si vous utilisez un ensemble ou une sélection temporaire dans un trigger, vous travaillez alors avec ceux de la machine où les triggers s'exécutent.
• **Interface utilisateur** : N'utilisez PAS d'éléments d'interface utilisateur dans un trigger (alerte, message ou dialogue). Cela signifie également que tracer le trigger dans la fenêtre du Débogueur doit être limité. Souvenez-vous que les triggers en client/serveur s'exécutent sur la machine 4D Server. Un message d'alerte affiché sur le poste serveur ne dit pas grand chose à l'utilisateur qui, lui, travaille sur sa machine cliente. Laissez le process appelant gérer l'interface utilisateur.

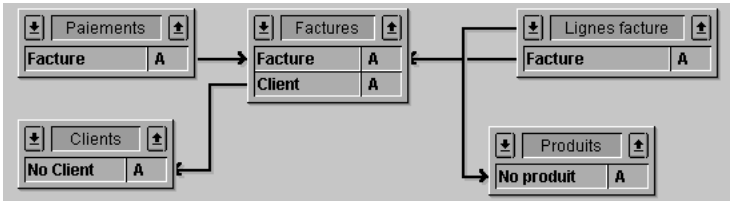
Triggers et transactions

Les transactions doivent être gérées au niveau du process appelant. Il est fortement déconseillé de gérer des transactions au niveau du trigger. Si, pendant l'exécution d'un trigger, vous devez ajouter, modifier ou détruire plusieurs enregistrements et souhaitez garantir l'intégrité de vos données à l'aide d'une transaction, vous devez d'abord tester (à partir du trigger) si le process appelant est en cours de transaction avec la commande Transaction en cours. En effet, si ce n'est pas le cas et si le trigger rencontre un enregistrement verrouillé, le process appelant n'aura aucun moyen d'annuler a posteriori les actions déjà effectuées par le trigger. Par conséquent, si vous n'êtes pas en transaction, ne commencez pas les opérations à exécuter, et retournez simplement une erreur dans $0 afin de signaler au process appelant que l'opération de base de données doit être exécutée dans une transaction.

Note : Afin d'optimiser le fonctionnement combiné des triggers et des transactions, 4D n'appelle PAS les triggers lors d'un VALIDER TRANSACTION. Cela évite que les triggers soient exécutés deux fois.

Triggers en cascade

Prenons l'exemple de la structure suivante :



Note : Les tables ont été contractées (il y a davantage de champs).

Pour les nécessités de cette documentation, nous admettrons que la base “autorise” la suppression d'une facture. Voyons aussi comment une telle opération serait gérée au niveau du trigger (puisque vous pourriez aussi décider d'effectuer l'opération au niveau du process).

Afin que soit maintenue l'intégrité relationnelle des données, la suppression d'une facture requiert les actions suivantes de la part du trigger de [Factures] :
• Décrémenter le champ Ventes de la table [Clients] du montant de la facture.
• Supprimer tous les enregistrements de [Lignes facture] liés à la facture.
• Ceci implique aussi que le trigger de [Lignes facture] décrémente le champ Quantité vendue des enregistrements [Produits] liés à la ligne de facture que l'on s'apprête à supprimer.
• Supprimer tous les enregistrements de [Paiements] liés à la facture.

Tout d'abord, le trigger de [Factures] ne doit effectuer ces actions que si le process appelant est en transaction, afin qu'une annulation rétroactive soit possible en cas de rencontre d'un enregistrement verrouillé.

Deuxièmement, le trigger de [Lignes facture] est **en cascade** avec le trigger de [Factures]. Le premier s'exécute “à l'intérieur” du second parce que la destruction des éléments de la liste est consécutive à un appel à SUPPRIMER SELECTION dans le trigger de [Factures].

Ajoutons que toutes les tables dans cet exemple ont des triggers activés pour tous les événements de la base de données. La cascade des triggers sera :

- Le trigger de Factures est appelé car le process appelant supprime une facture
  - Le trigger de Clients est appelé car le trigger Factures met à jour le champ Ventes
  - Le trigger de Lignes facture est appelé car le trigger Factures supprime une ligne (ce qui est répété)
    - Le trigger de Produits est appelé car le trigger Lignes facture met à jour le champ Quantité vendue
  - Le trigger de Paiements est appelé car le trigger Factures supprime un paiement (ce qui est répété)

Dans cette cascade, le trigger de [Factures] s'exécute au niveau 1, les triggers [Clients], [Lignes facture] et [Paiements] au niveau 2 et le trigger [Produits] au niveau 3.

Dans les triggers, vous pouvez détecter à quel niveau un trigger est exécuté grâce à la commande Niveau du trigger. De plus, vous pouvez aussi obtenir des informations sur les autres niveaux en utilisant la commande PROPRIETES DU TRIGGER.

Si, par exemple, vous détruisiez un enregistrement [Produits] à un niveau process, le trigger de [Produits] s'exécuterait au niveau 1, non au niveau 3, comme plus haut.

Avec Niveau du trigger et PROPRIETES DU TRIGGER, vous pouvez identifier la raison d'une action. Dans l'exemple ci-dessus, une facture est supprimée au niveau process. Prenons pour hypothèse que nous voulons détruire un enregistrement [Clients] au niveau process. Le trigger de [Clients] devrait alors être conçu pour détruire toutes les factures liées à ce client. Cela signifie que le trigger [Factures] devrait être invoqué comme plus haut, mais pour une autre raison. Du trigger [Factures], vous pouvez détecter si le niveau est 1 ou 2. S'il est 2, vous pouvez vérifier si oui ou non c'est à cause de la suppression de l'enregistrement Client lui-même. Si tel est le cas, vous n'avez même plus besoin de vous préoccupper de la mise à jour du champ Ventes.

Utilisation de la numérotation automatique dans un trigger

Quand vous manipulez l'événement moteur Sur sauvegarde nouvel enreg, vous pouvez appeler la commande Numerotation automatique pour maintenir un numéro d'identification unique pour chaque enregistrement d'une table. Par exemple :

```
   ` Trigger pour la table [Factures]
Au cas ou
   : (Evenement moteur=Sur sauvegarde nouvel enreg)
      ` ...
      [Factures]Facture Identification:=Numerotation automatique ([Factures])
      ` ...
Fin de cas
```

Référence

Evenement moteur, Méthodes, Niveau du trigger, Numero enregistrement, PROPRIETES DU TRIGGER.

Evenement moteur → Entier long

Paramètre	Type			Description
Cette commande ne requiert pas de paramètre				
Résultat	Entier long	←	0	Hors de tout événement de trigger
			1	Sauvegarde d'un nouvel enregistrement
			2	Sauvegarde d'un enregistrement existant
			3	Suppression d'un enregistrement
			4	Chargement d'un enregistrement

Description

La commande Evenement moteur est appelée dans un trigger et renvoie une valeur numérique qui indique le type de l'événement de la base, ou la raison pour laquelle le trigger a été appelé.

4D fournit les constantes prédéfinies suivantes :

Constante	Type	Valeur
Sur sauvegarde nouvel enreg	Entier long	1
Sur sauvegarde enregistrement	Entier long	2
Sur suppression enregistrement	Entier long	3
Sur chargement enregistrement	Entier long	4

Si, dans un trigger, vous effectuez des opérations de base de données sur plusieurs enregistrements (par exemple mise à jour de plusieurs enregistrements dans la table [Produits] et ajout d'enregistrement dans la table [Factures]), vous pouvez rencontrer des situations (comme des enregistrements verrouillés) qui empêchent le trigger d'exécuter correctement les opérations pour lesquelles il est appelé. Il vous faut alors stopper les actions de la base et retourner une erreur pour que le process appelant sache que la requête n'a pu être exécutée. Ce process doit également être en mesure d'annuler les opérations non exécutées. Autrement dit, lorsqu'une telle situation se produit, vous avez besoin de savoir dans le trigger si vous êtes en transaction avant même d'essayer de faire quoi que ce soit. Pour cela, utilisez la fonction Transaction en cours.

Dans 4e Dimension, il n'y a pas de limite, à part la mémoire disponible, aux appels de triggers en cascade. Pour optimiser l'exécution d'un trigger, vous pouvez écrire le code de vos triggers non seulement en fonction de l'événement de la base mais aussi du niveau de l'appel lorsque les triggers sont appelés en cascade. Par exemple, pendant l'événement de la base Suppression enregistrement pour la table [Factures], vous pouvez ne pas effectuer la mise à jour du champ [Clients]Ventes si la suppression de l'enregistrement de la table [Factures] fait partie de la suppression en cascade des factures liées à l'enregistrement dans la table [Clients] que vous êtes en train de supprimer. Pour cela, utilisez les routines Niveau du trigger et PROPRIETES DU TRIGGER.

Exemple

Utilisez la fonction Evenement moteur pour structurer vos triggers comme ci-dessous :

```
   ` Un trigger de la table [toute table]

   C_ENTIER LONG($0)
   $0:=0   ` S'assurer que la requête de la base sera accordée
   Au cas ou
⇒     : (Evenement moteur=Sur sauvegarde nouvel enreg)
         ` Exécuter les actions appropriées pour la sauvegarde d'un nouvel enregistrement
⇒     : (Evenement moteur=Sur sauvegarde enregistrement)
         ` Exécuter les actions appropriées pour la sauvegarde d'un enregistrement existant
⇒     : (Evenement moteur=Sur suppression enregistrement)
         ` Exécuter les actions appropriées pour la suppression d'un enregistrement
⇒     : (Evenement moteur=Sur chargement enregistrement)
         ` Exécuter les actions appropriées pour le chargement d'un enregistrement en
                                                                          mémoire
   Fin de cas
```

Référence

Niveau du trigger, Présentation des triggers, PROPRIETES DU TRIGGER, Transaction en cours.

Niveau du trigger

Niveau du trigger → Numérique

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Numérique	←	Niveau d'exécution du trigger (0 si hors du cycle d'exécution du trigger)

Description

La commande Niveau du trigger retourne le niveau d'exécution du trigger.

Reportez-vous à la description des triggers en cascade dans la section Présentation des triggers.

Référence

Evenement moteur, Présentation des triggers, PROPRIETES DU TRIGGER.

PROPRIETES DU TRIGGER (niveauTrigger; evenementBase; tableNum; enregNum)

Paramètre	Type		Description
niveauTrigger	Numérique	→	Niveau d'exécution du trigger
evenementBase	Numérique	←	Evénement de base de données
tableNum	Numérique	←	Numéro de la table
enregNum	Numérique	←	Numéro de l'enregistrement

Description

La commande PROPRIETES DU TRIGGER fournit des informations sur le niveau d'exécution du trigger que vous avez passé dans niveauTrigger. Vous devez utiliser conjointement PROPRIETES DU TRIGGER et Niveau du trigger pour effectuer différentes actions en fonction de la cascade du trigger. Reportez-vous à la description des triggers en cascade dans la section Présentation des triggers.

Si vous passez un niveau d'exécution de trigger inexistant, la commande retourne 0 (zéro) dans chaque paramètre.

La nature de l'événement de base de données pour le niveau d'exécution du trigger est retournée dans evenementBase. Les constantes prédéfinies suivantes sont fournies :

Constante	Type	Valeur
Sur sauvegarde nouvel enreg	Entier long	1
Sur sauvegarde enregistrement	Entier long	2
Sur suppression enregistrement	Entier long	3
Sur chargement enregistrement	Entier long	4

Le numéro de table et d'enregistrement pour l'enregistrement concerné par l'événement de base de données pour le niveau d'exécution du trigger sont retournés dans tableNum et enregNum.

Note : Rappelez-vous que, pendant une transaction, les numéros des enregistrements nouvellement créés sont des numéros temporaires.

Référence

A propos des numéros d'enregistrements, Evenement moteur, Niveau du trigger, Présentation des triggers.

55

Utilisateurs et groupes

CHANGER PRIVILEGES

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

CHANGER PRIVILEGES permet de modifier le système de mots de passe. Lorsque cette commande est exécutée, la fenêtre des mots de passe du mode Structure est appelée pour modifier les privilèges.

Les groupes peuvent être modifiés par le Super_Utilisateur et l'Administrateur et par les propriétaires de groupe. Seuls le Super_Utilisateur et l'Administrateur peuvent modifier tous les groupes. Les propriétaires de groupe ne peuvent modifier que leur propre groupe. Des utilisateurs peuvent être ajoutés et retirés des groupes. Cette commande ne fait rien si aucun groupe n'est défini.

Le Super_Utilisateur et l'Administrateur peuvent créer des utilisateurs et les placer dans des groupes.
Dans le cadre d'une base client-serveur, le Super_Utilisateur et l'Administrateur peuvent également utiliser cette commande pour enregistrer l'accès personnalisé à la base 4D Server (voir la section Enregistrer l'accès à la base 4D Server dans le manuel de référence de 4D Server).

Exemple

L'exemple suivant affiche la fenêtre des mots de passe :

```
⇒ CHANGER PRIVILEGES
```

Référence

CHANGER MOT DE PASSE, CHANGER UTILISATEUR.

CHANGER UTILISATEUR

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

CHANGER UTILISATEUR permet à l'utilisateur de changer de privilèges d'accès sans quitter la base, en choisissant un autre nom d'utilisateur et en saisissant son mot de passe. La boîte de dialogue de connexion à la base se présente et l'utilisateur peut entrer dans la base sous un autre nom d'utilisateur. Lorsqu'il change d'identité, il abandonne ses anciens privilèges au profit de ceux de l'utilisateur choisi.

La boîte de dialogue qui s'affiche dépend des options fixées dans la fenêtre des Propriétés de la base, en mode Structure.

Exemple

L'exemple suivant affiche la boîte de dialogue de connexion :

```
⇒ CHANGER UTILISATEUR
```

Référence

CHANGER MOT DE PASSE.

Valider mot de passe (réfUtilisateur; motDePasse) → Booléen

Paramètre	Type		Description
réfUtilisateur	Numérique	→	N° de référence unique
motDePasse	Alpha	→	Mot de passe non crypté
Résultat	Booléen	←	Vrai = mot de passe correct Faux = mot de passe incorrect

Description

La commande Valider mot de passe retourne Vrai si la chaîne passée dans motDePasse est le mot de passe du compte utilisateur dont le n° de référence est passé dans réfUtilisateur.

Exemple

L'exemple suivant vérifie que "Laurel" est le mot de passe de l'utilisateur "Hardy" :

```
LIRE LISTE UTILISATEURS(atNomUtil;alNumérosUtil)
$vlElem:=Chercher dans tableau(atNomUtil;"Hardy")
Si ($vlElem>0)
⇒    Si (Valider mot de passe(alRefUtil{$vlElem} ;"Laurel")>0)
        ALERTE("Oui !")
     Sinon
        ALERTE("Dommage !")
     Fin de si
Sinon
     ALERTE("Nom d'utilisateur inconnu")
Fin de si
```

Référence

Ecrire proprietes utilisateur, LIRE PROPRIETES UTILISATEUR.

CHANGER MOT DE PASSE (motDePasse)

Paramètre	Type		Description
motDePasse	Alpha	→	Nouveau mot de passe

Description

CHANGER MOT DE PASSE permet de changer le mot de passe de l'utilisateur courant. Cette commande remplace le mot de passe courant par le nouveau mot de passe que vous passez dans motDePasse.

Attention : Les mots de passe différencient les caractères majuscules et minuscules.

Exemple

L'exemple suivant permet à l'utilisateur de modifier son mot de passe :

```
CHANGER UTILISATEUR  ` Afficher la boîte de dialogue des mots de passe
Si (OK=1)
   $pw1:=Demander("Saisissez le nouveau mot de passe pour "+Utilisateur courant)
      ` Le mot de passe doit comporter au moins cinq caractères
   Si (((OK=1) & ($pw1#"")) & (Longueur($pw1)>5))
         ` Vérifier qu'un mot de passe valide a été saisi
      $pw2:=Demander("Saisissez de nouveau le mot de passe")
      Si ((OK=1) & ($pw1=$pw2))
⇒        CHANGER MOT DE PASSE($pw2)  ` Modifier le mot de passe
      Fin de si
   Fin de si
Fin de si
```

Référence

CHANGER UTILISATEUR.

Utilisateur courant → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Nom de l'utilisateur courant

Description

Utilisateur courant retourne le "nom d'utilisateur" de l'utilisateur courant.

Exemple

Reportez-vous à l'exemple de la commande Appartient au groupe.

Référence

Appartient au groupe, CHANGER MOT DE PASSE, CHANGER UTILISATEUR.

Appartient au groupe (nomUtilisateur; groupe) → Booléen

Paramètre	Type		Description
nomUtilisateur	Alpha	→	Nom de l'utilisateur
groupe	Alpha	→	Nom du groupe
Résultat	Booléen	←	Vrai = utilisateur est dans groupe Faux = utilisateur n'est pas dans groupe

Description

La fonction Appartient au groupe retourne Vrai si utilisateur appartient au groupe.

Exemple

L'exemple suivant recherche des factures. Si l'utilisateur courant est dans le groupe Administration, il pourra accéder aux formulaires qui affichent des informations confidentielles. Sinon, des formulaires standard sont affichés :

```
      CHERCHER([Factures];[Factures]Prix>100)
⇒     Si (Appartient au groupe(Utilisateur courant;"Administration")
         FORMULAIRE SORTIE([Factures];"Confidentiel_Sortie")
         FORMULAIRE ENTREE([Factures];"Conf_Saisie")
      Sinon
         FORMULAIRE SORTIE([Factures];"Sortie_Standard")
         FORMULAIRE ENTREE([Factures];"Entrée_Standard")
      Fin de si
      MODIFIER SELECTION([Factures];*)
```

Référence

Utilisateur courant.

SUPPRIMER UTILISATEUR (réfUtilisateur)

Paramètre	Type		Description
réfUtilisateur	Numérique	→	Numéro d'identification de l'utilisateur à supprimer

Description

La commande SUPPRIMER UTILISATEUR supprime l'utilisateur dont le numéro est passé dans réfUtilisateur. Vous devez passer un numéro valide d'utilisateur, retourné par la commande LIRE LISTE UTILISATEURS.

Si le compte de l'utilisateur n'existe pas ou a déjà été supprimé, une erreur -9979 est genérée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Les utilisateurs supprimés n'apparaissent plus dans la fenêtre des mots de passe qui est affichée lorsque vous appelez CHANGER UTILISATEUR. Cependant, afin de maintenir des numéros d'utilisateur uniques, le compte de l'utilisateur est conservé dans le système de mots de passe. Les utilisateurs supprimés sont affichés en vert dans la fenêtre des mots de passe en mode Structure.

Référence

ECRIRE PROPRIETES UTILISATEUR, LIRE LISTE UTILISATEURS, LIRE PROPRIETES UTILISATEUR, Utilisateur supprime.

Gestion des erreurs

Si vous n'avez pas les privilèges d'accès pour appeler SUPPRIMER UTILISATEUR ou si un autre process accède déjà au système de mots de passe, une erreur de privilège d'accès est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Utilisateur supprime

Utilisateur supprime (réfUtilisateur) → Booléen

Paramètre	Type		Description
réfUtilisateur	Numérique	→	Numéro d'identification de l'utilisateur
Résultat	Booléen	←	Vrai = le compte de l'utilisateur est supprimé ou n'existe pas Faux = le compte de l'utilisateur est actif

Description

La commande Utilisateur supprime teste le compte de l'utilisateur dont le numéro d'identification unique est passé dans réfUtilisateur.

Si le compte n'existe pas ou a été supprimé, la fonction Utilisateur supprime retourne Vrai. Sinon, elle retourne Faux.

Référence

ECRIRE PROPRIETES UTILISATEUR, LIRE PROPRIETES UTILISATEUR, SUPPRIMER UTILISATEUR.

Gestion des erreurs

Si vous n'avez pas les privilèges d'accès pour appeler Utilisateur supprime ou si un autre process accède déjà au système de mots de passe, une erreur de privilège d'accès est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs utilisant la commande APPELER SUR ERREUR.

LIRE LISTE UTILISATEURS (nomsUtil; numérosUtil)

Paramètre	Type		Description
nomsUtil	Tableau alpha	←	Noms des utilisateurs tels qu'ils apparaissent dans l'éditeur de Mots de passe
réfsUtil	Tableau num	←	Numéros de référence uniques pour chaque utilisateur

Description

La commande LIRE LISTE UTILISATEURS remplit les tableaux nomsUtil et réfsUtil avec les noms et les numéros de référence uniques des utilisateurs tels qu'ils apparaissent dans la fenêtre des Mots de passe de 4D.

Le tableau nomsUtil est rempli avec les noms des utilisateurs, y compris ceux dont le compte est supprimé (les utilisateurs dont le nom apparaît en vert dans la fenêtre des mots de passe).

Note : Utilisez la commande Utilisateur supprime pour savoir si un compte utilisateur est supprimé.

Le tableau réfsUtil, synchronisé avec nomsUtil, est rempli avec les numéros de référence uniques des utilisateurs. Ces numéros peuvent être les suivants :

Numéro de référence de l'utilisateur	Description utilisateur
1	Super_Utilisateur
2	Administrateur
1 à 15000	Utilisateur créé par le Super_Utilisateur (l'utilisateur n°3 est le 1er utilisateur créé par le Super_Utilisateur, l'utilisateur n°4 est le second, et ainsi de suite).
-11 à -15010	Utilisateur créé par l'Administrateur (l'utilisateur n°-11 est le 1er utilisateur créé par l'Administrateur, l'utilisateur n°-12 est le second, et ainsi de suite).

Référence

Ecrire proprietes utilisateur, LIRE LISTE GROUPE, LIRE PROPRIETES UTILISATEUR.

Gestion des erreurs

Si vous n'avez pas les privilèges d'accès pour appeler la commande LIRE LISTE UTILISATEURS ou si le système des Mots de passe est déjà ouvert par un autre process, une erreur de privilège d'accès est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par APPELER SUR ERREUR.

LIRE PROPRIETES UTILISATEUR (réfUtilisateur; nom; démarrage; motDePasse; nbUtilisations; dernièreUtilisation{; adhésions})

Paramètre	Type		Description
réfUtilisateur	Numérique	→	Numéro de référence unique de l'utilisateur
nom	Alpha	←	Nom de l'utilisateur
démarrage	Alpha	←	Nom de la méthode de démarrage
motDePasse	Alpha	←	Chaîne vide
nbUtilisations	Numérique	←	Nombre d'utilisations de la base
dernièreUtilisation	Date	←	Date de la dernière utilisation de la base
adhésions	Tableau num	←	Numéros de référence des groupes auxquels l'utilisateur appartient

Description

LIRE PROPRIETES UTILISATEUR retourne les informations concernant l'utilisateur dont le numéro de référence est passé dans le paramètre réfUtilisateur. Vous devez passer le numéro de référence retourné par la commande LIRE LISTE UTILISATEURS.

Si le compte d'utilisateur n'existe pas ou a été supprimé, l'erreur -9979 est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par APPELER SUR ERREUR. Sinon, vous pouvez appeler la fonction Utilisateur supprime pour tester le compte de l'utilisateur avant d'appeler LIRE PROPRIETES UTILISATEUR.

Les numéros de référence pour les utilisateurs peuvent être les suivants :

Numéro de référence de l'utilisateur	Description utilisateur
1	Super_Utilisateur
2	Administrateur
1 à 15000	Utilisateur créé par le Super_Utilisateur (l'utilisateur n°3 est le 1er utilisateur créé par le Super_Utilisateur, l'utilisateur n°4 est le second, et ainsi de suite).
-11 à -15010	Utilisateur créé par l'Administrateur (l'utilisateur n°-11 est le 1er utilisateur créé par l'Administrateur, l'utilisateur n°-12 est le second, et ainsi de suite).

Après l'appel, pour chaque utilisateur dont la référence est réfUtilisateur, vous récupérez aussi le nom, la méthode de démarrage, le nombre d'utilisations et la date de la dernière utilisation de la base dans les paramètres nom, démarrage, nbUtilisation et dernièreUtilisation.

Note : La commande LIRE PROPRIETES UTILISATEUR ne retourne plus le mot de passe crypté dans le paramètre motDePasse. Depuis la version 6.0.2 de 4D, une chaîne vide est toujours retournée dans ce paramètre. Si vous souhaitez contrôler le mot de passe d'un utilisateur, utilisez la fonction Valider mot de passe.

Si vous passez le paramètre optionnel adhésion, vous récupérez le numéro de référence unique du groupe auquel l'utilisateur appartient. Ces numéros sont les suivants :

Numéro de référence du groupe	Description du groupe
15001 à 32767	Groupe créé par le Super_Utilisateur ou le propriétaire du groupe (le groupe n°15001 est le 1er groupe créé par le Super_Utilisateur, le groupe n°15002 est le second, etc.
-15011 à -32768	Groupe créé par l'Administrateur ou le propriétaire du groupe (le groupe n°-15001 est le 1er groupe créé par l'Administrateur, le groupe n°-15002 est le second, etc.

Référence

ECRIRE PROPRIETES UTILISATEUR, LIRE LISTE GROUPE, LIRE LISTE UTILISATEURS, Valider mot de passe.

Gestion des erreurs

Si vous n'avez pas les privilèges d'accès pour appeler la commande LIRE PROPRIETES UTILISATEUR ou si le système de Mots de passe est déjà ouvert par un autre process, une erreur de privilège d'accès est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par APPELER SUR ERREUR.

Ecrire proprietes utilisateur (réfUtilisateur; nom; démarrage; motDePasse; nbUtilisation; dernièreUtilisation{; adhésions}) → Numérique

Paramètre	Type		Description
réfUtilisateur	Numérique	→	Numéro de référence unique du compte de l'utilisateur ou -1 pour l'ajout d'un utilisateur affilié au Super_Utilisateur ou -2 pour l'ajout d'un utilisateur affilié à l'Administrateur
nom	Alpha	→	Nouveau nom de l'utilisateur
démarrage	Alpha	→	Nom de la nouvelle méthode de démarrage
motDePasse	Alpha	→	Nouveau mot de passe (non crypté) ou * pour ne pas modifier le mot de passe
nbUtilisation	Numérique	→	Nouveau nombre d'utilisations de la base
dernièreUtilisation	Date	→	Nouvelle date de dernière utilisation de la base
adhésions	Tableau num	→	Numéros de référence des groupes auxquels l'utilisateur appartient
Résultat	Numérique	←	Numéro de référence unique du nouvel utilisateur

Description

Ecrire proprietes utilisateur vous permet de modifier et de mettre à jour les propriétés d'un compte actif d'utilisateur existant dont le numéro de référence est passé dans le paramètre réfUtilisateur, ou d'ajouter un nouvel utilisateur affilié soit au Super_Utilisateur soit à l'Administrateur.

Si vous modifiez les propriétés d'un utilisateur existant, vous devez passer le numéro de référence qui vous est renvoyé par la commande LIRE LISTE UTILISATEURS.

Si le compte d'utilisateur n'existe pas ou a été supprimé, Ecrire proprietes utilisateur retourne 0 et l'erreur -9979 est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par APPELER SUR ERREUR. Sinon, vous pouvez appeler la fonction Utilisateur supprime pour tester le compte de l'utilisateur avant d'appeler Ecrire proprietes utilisateur.

Les numéros de référence pour les utilisateurs peuvent être les suivants :

Numéro de référence de l'utilisateur	Description utilisateur
1	Super_Utilisateur
2	Administrateur
1 à 15000	Utilisateur créé par le Super_Utilisateur (l'utilisateur n°3 est le 1er utilisateur créé par le Super_Utilisateur, l'utilisateur n°4 est le second, et ainsi de suite).
-11 à -15000	Utilisateur créé par l'Administrateur (l'utilisateur n°-11 est le 1er utilisateur créé par l'Administrateur, l'utilisateur n°-12 est le second, et ainsi de suite).

Si vous voulez ajouter un nouvel utilisateur affilié au Super_Utilisateur, il faut passer -1 à réfUtilisateur. Si vous voulez ajouter un nouvel utilisateur affilié à l'Administrateur, il faut passer -2 à réfUtilisateur.

Si l'utilisateur a bien été créé, Ecrire proprietes utilisateur retourne son numéro de référence unique d'utilisateur.
Si vous ne passez pas un numéro de référence d'utilisateur valide, Ecrire proprietes utilisateur ne fait rien et retourne 0.

Lorsque vous appelez cette commande, vous passez le nouveau nom, la nouvelle méthode de démarrage, le nouveau mot de passe, le nouveau nombre d'utilisations et la nouvelle date de dernière utilisation pour l'utilisateur dans les paramètres nom, démarrage, motDePasse, nbUtilisation et dernièreUtilisation. Vous passez un mot de passe non crypté dans le paramètre motDePasse. 4D cryptera ce mot de passe avant de le sauvegarder dans le compte de l'utilisateur.
Si le nouveau nom d'utilisateur passé dans nom n'est pas unique (un utilisateur de même nom existe déjà), la commande ne fait rien et l'erreur -9979 est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par APPELER SUR ERREUR.

Si vous ne voulez pas modifier toutes les propriétés de l'utilisateur (à part son groupe, voir ci-dessous), appelez au préalable LIRE PROPRIETES UTILISATEUR et passez les valeurs retournées dans celles que vous ne voulez pas modifier. Si vous ne voulez pas modifier le mot de passe de l'utilisateur, passez * dans le paramètre motDePasse. Cela vous permet de changer les autres propriétés du compte de l'utilisateur, sans changer le mot de passe de ce compte.

Si vous ne passez pas le paramètre optionnel adhésions, les adhésions de l'utilisateur restent inchangées. Si vous ne passez pas ce paramètre en cas d'ajout d'un utilisateur, il ne fera partie d'aucun groupe.

Si vous passez le paramètre optionnel adhésions, vous modifiez toutes les adhésions pour l'utilisateur. Avant d'appeler cette commande, vous devez remplir le tableau adhésions avec les numéros de référence uniques des groupes dont l'utilisateur devra faire partie. Les numéros de référence des groupes sont les suivants :

Numéro de référence du groupe	Description du groupe
15001 à 32767	Groupe créé par le Super_Utilisateur ou le propriétaire du groupe (le groupe n°15001 est le 1er groupe créé par le Super_Utilisateur, le groupe n°15002 est le second, etc.
-15001 à -32768	Groupe créé par l'Administrateur ou le propriétaire du groupe (le groupe n°-15001 est le 1er groupe créé par l'Administrateur, le groupe n°-15002 est le second, etc.

Si vous voulez annuler les adhésions d'un utilisateur, passez un tableau vide dans le paramètre adhésion.

Référence

LIRE LISTE GROUPE, LIRE LISTE UTILISATEURS, LIRE PROPRIETES UTILISATEUR, SUPPRIMER UTILISATEUR, Utilisateur supprime, Valider mot de passe.

Gestion des erreurs

Si vous n'avez pas les privilèges d'accès pour appeler Ecrire proprietes utilisateur ou si le système de mots de passe est déjà ouvert par un autre process, une erreur de privilège d'accès est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par APPELER SUR ERREUR.

LIRE LISTE GROUPE (nomsGroupe; numérosGroupe)

Paramètre	Type		Description
nomsGroupe	Tableau alpha	←	Noms des groupes tels qu'ils apparaissent dans l'éditeur de Mots de passe
numérosGroupe	Tableau num	←	Numéros de référence uniques pour chaque groupe

Description

LIRE LISTE GROUPE remplit les tableaux nomsGroupe et numérosGroupe avec les noms et les numéros de référence uniques des groupes tels qu'ils apparaissent dans l'éditeur de Mots de passe.

Le tableau numérosGroupe, synchronisé avec le tableau nomsGroupe, est rempli avec les numéros de référence uniques des groupes. Ces numéros sont les suivants :

Numéro de référence du groupe	Description du groupe
15001 à 32767	Groupe créé par le Super_Utilisateur ou le propriétaire du groupe (le groupe n°15001 est le 1er groupe créé par le Super_Utilisateur, le groupe n°15002 est le second, etc.
-15011 à -32768	Groupe créé par l'Administrateur ou le propriétaire du groupe (le groupe n°-15001 est le 1er groupe créé par l'Administrateur, le groupe n°-15002 est le second, etc.

Référence

Ecrire proprietes groupe, LIRE LISTE UTILISATEURS, LIRE PROPRIETES GROUPE.

Gestion des erreurs

Si vous n'avez pas les privilèges d'accès pour appeler la commande LIRE LISTE GROUPE ou si le système de Mots de passe est déjà ouvert par un autre process, une erreur de privilège d'accès est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par APPELER SUR ERREUR.

LIRE PROPRIETES GROUPE (réfGroupe; nom; propriétaire{; membres})

Paramètre	Type		Description
réfGroupe	Numérique	→	Numéro de référence du groupe
nom	Alpha	←	Nom du groupe
propriétaire	Numérique	←	Numéro de référence du propriétaire du groupe
membres	Tableau num	←	Membres du groupe

Description

LIRE PROPRIETES GROUPE retourne les propriétés du groupe dont le numéro de référence est passé dans réfGroupe. Vous passez le numéro de référence du groupe retourné par la commande LIRE LISTE GROUPE. Les numéros de référence des groupes sont les suivants :

Numéro de référence du groupe	Description du groupe
15001 à 32767	Groupe créé par le Super_Utilisateur ou le propriétaire du groupe (le groupe n°15001 est le 1er groupe créé par le Super_Utilisateur, le groupe n°15002 est le second, etc.
-15011 à -32768	Groupe créé par l'Administrateur ou le propriétaire du groupe (le groupe n°-15001 est le 1er groupe créé par l'Administrateur, le groupe n°-15002 est le second, etc.

Si vous ne passez pas un numéro de référence valide, LIRE PROPRIETES GROUPE renvoie des paramètres vides.

Après l'appel de la commande, vous récupérez le nom et le numéro du propriétaire du groupe dans les paramètres nom et propriétaire.

Si vous passez le paramètre optionnel membres, ce tableau contiendra les numéros de référence uniques des utilisateurs qui appartiennent au groupe.

Les numéros de référence des membres de groupe sont les suivants :

Numéro de référence du membre	Description membre
1	Super_Utilisateur
2	Administrateur
1 à 15000	Utilisateur créé par le Super_Utilisateur (l'utilisateur n°3 est le 1er utilisateur créé par le Super_Utilisateur, l'utilisateur n°4 est le second, et ainsi de suite).
-11 à -15010	Utilisateur créé par l'Administrateur (l'utilisateur n°-11 est le 1er utilisateur créé par l'Administrateur, l'utilisateur n°-12 est le second, et ainsi de suite).
15001 à 32767	Groupe créé par le Super_Utilisateur ou le propriétaire du groupe (le groupe n°15001 est le 1er groupe créé par le Super_Utilisateur, le groupe n°15002 est le second, etc.
-15011 à -32768	Groupe créé par l'Administrateur ou le propriétaire du groupe (le groupe n°-15001 est le 1er groupe créé par l'Administrateur, le groupe n°-15002 est le second, etc.

Référence

ECRIRE PROPRIETES GROUPE, LIRE LISTE GROUPE, LIRE LISTE UTILISATEURS.

Gestion des erreurs

Si vous n'avez pas les privilèges d'accès pour appeler la commande LIRE PROPRIETES GROUPE ou si le système de Mots de passe est déjà ouvert par un autre process, une erreur de privilège d'accès est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par APPELER SUR ERREUR.

Ecrire proprietes groupe (réfGroupe; nom; propriétaire{; membres}) → Numérique

Paramètre	Type		Description
réfGroupe	Numérique	→	Numéro de référence unique du groupe activé ou -1 pour ajouter un groupe de Super_Utilisateur -2 pour ajouter un groupe d'Administrateur
nom	Alpha	→	Nouveau nom de groupe
propriétaire	Numérique	→	Numéro de référence unique de l'utilisateur ou le propriétaire du nouveau groupe
membres	Tableau num	→	Nouveaux membres du groupe
Résultat	Numérique	←	Numéro de référence unique du nouveau groupe

Description

Ecrire proprietes groupe vous permet de modifier et de mettre à jour les propriétés d'un groupe existant dont vous passez le numéro de référence unique dans réfGroupe, ou d'ajouter un nouveau groupe affilié au Super_Utilisateur ou à l'Administrateur.

Si vous modifiez les propriétés d'un groupe existant, vous devez passer son numéro de référence tel que retourné dans la commande LIRE LISTE GROUPE. Les numéros de référence de groupe sont les suivants :

Numéro de référence du groupe	Description du groupe
15001 à 32767	Groupe créé par le Super_Utilisateur ou le propriétaire du groupe (le groupe n°15001 est le 1er groupe créé par le Super_Utilisateur, le groupe n°15002 est le second, etc.
-15011 à -32768	Groupe créé par l'Administrateur ou le propriétaire du groupe (le groupe n°-15001 est le 1er groupe créé par l'Administrateur, le groupe n°-15002 est le second, etc.

Si vous voulez ajouter un nouveau groupe affilié au Super_Utilisateur, passez -1 dans réfGroupe. Si vous voulez ajouter un nouveau groupe affilié à l'Administrateur, passez -2 dans réfGroupe.

Si le groupe a bien été créé, Ecrire proprietes groupe retourne son numéro de référence unique.

Si vous ne passez pas -1, -2 ou un numéro de référence de groupe valide, Ecrire proprietes groupe ne fait rien et retourne 0.

Avant d'appeler cette routine, vous passez le nouveau nom du groupe et le numéro du propriétaire du groupe dans les paramètres nom et propriétaire. Si vous ne voulez pas modifier toutes les propriétés du groupe (à part ses membres, voir ci-dessous), passez les valeurs retournées par LIRE PROPRIETES GROUPE dans les paramètres que vous voulez laisser inchangés.

Si vous ne passez pas le paramètre optionnel membres, la liste courante des membres du groupe reste inchangée. Si vous le faites lors d'une création d'un groupe, le groupe n'aura pas de membres.

Note : Le propriétaire d'un groupe n'est pas automatiquement défini comme membre du groupe qu'il possède. C'est à vous de l'y inclure explicitement, à l'aide du paramètre membres.

Si vous passez le paramètre optionnel membres, vous modifiez toute la liste des membres pour ce groupe. Avant d'appeler cette routine, vous devez remplir le tableau membres avec les numéros de référence uniques des utilisateurs et/ou des groupes devant appartenir au groupe. Ces numéros peuvent être les suivants :

Numéro de référence du membre	Description membre
1	Super_Utilisateur
2	Administrateur
1 à 15000	Utilisateur créé par le Super_Utilisateur (l'utilisateur n°3 est le 1er utilisateur créé par le Super_Utilisateur, l'utilisateur n°4 est le second, et ainsi de suite).
-11 à -15010	Utilisateur créé par l'Administrateur (l'utilisateur n°-11 est le 1er utilisateur créé par l'Administrateur, l'utilisateur n°-12 est le second, et ainsi de suite).
15001 à 32767	Groupe créé par le Super_Utilisateur ou le propriétaire du groupe (le groupe n°15001 est le 1er groupe créé par le Super_Utilisateur, le groupe n°15002 est le second, etc.
-15011 à -32768	Groupe créé par l'Administrateur ou le propriétaire du groupe (le groupe n°-15001 est le 1er groupe créé par l'Administrateur, le groupe n°-15002 est le second, etc.

Si vous voulez enlever tous les membres d'un groupe, passez un tableau vide dans le paramètre membres.

Référence

LIRE LISTE GROUPE, LIRE LISTE UTILISATEURS, LIRE PROPRIETES GROUPE.

Gestion des erreurs

Si vous n'avez pas les privilèges d'accès pour appeler la commande Ecrire proprietes groupe ou si le système de mots de passe est déjà ouvert par un autre process, une erreur de privilège d'accès est générée. Vous pouvez intercepter cette erreur avec une méthode de gestion d'erreurs installée par APPELER SUR ERREUR.

CHANGER LICENCES

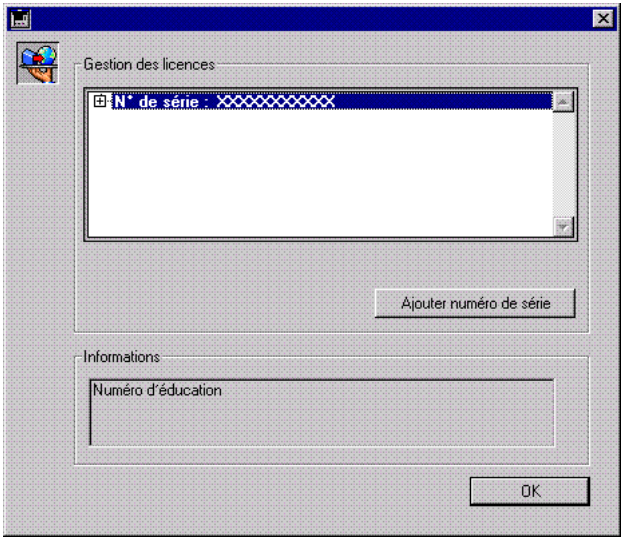
Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande CHANGER LICENCES affiche la boîte de dialogue de gestion des licences 4D. Cette boîte de dialogue vous permet d'ajouter des numéros de série (pour activer des plug-ins et le serveur Web).
En outre, avec 4D Server, cette boîte de dialogue vous permet d'ajouter des numéros d'expansion afin d'accroître le nombre de clients pouvant utiliser simultanément la base et ses plug-ins.

Voici la boîte de dialogue de gestion des licences sous Windows :



Note : En mode Structure, vous affichez cette boîte de dialogue en cliquant sur le bouton **Licences** de la boîte de dialogue **Propriétés** de la base de données.

CHANGER LICENCES permet d'activer des licences et d'ajouter des numéros d'expansion dans une application compilée diffusée à vos clients. Les développeurs 4D et les administrateurs de systèmes peuvent utiliser cette commande pour diffuser une application 4D, en laissant à leurs clients le soin de saisir eux-même les numéros sans devoir leur faire parvenir une mise à jour de l'application.

Pour plus d'informations sur le fonctionnement de cette boîte de dialogue, reportez-vous au *Guide d'installation* de la gamme 4D Product Line.

Exemple

Dans une boîte de dialogue de configuration ou de préférences personnalisée, vous placez un bouton auquel la méthode suivante est associée :

```
` Méthode objet du bouton bLicence
⇒ CHANGER LICENCES
```

Vous permettrez ainsi à l'utilisateur d'activer des licences sans avoir à modifier la base de données.

56

Variables

ECRIRE VARIABLES (doc; variable{; variable2; ...; variableN})

Paramètre	Type		Description
doc	Alpha	→	Nom du document dans lequel sauvegarder la ou les variable(s)
variable	Variable	→	Variable(s) à sauvegarder

Description

La commande ECRIRE VARIABLES sauvegarde une ou plusieurs variable(s) dans un document disque dont le nom est passé dans le paramètre doc.

Les variables ne doivent pas obligatoirement être du même type, mais doivent avoir le type Texte, Numérique, Date, Heure, Booléen ou Image.

Si vous passez une chaîne vide ("") dans doc, une boîte de dialogue standard d'enregistrement de fichiers apparaît, permettant à l'utilisateur de donner un nom au document à créer. Dans ce cas, la variable système Document récupère le nom du document, s'il a bien été créé.

Si les variables ont été correctement sauvegardées, la variable système OK prend la valeur 1. Sinon, OK prend la valeur 0.

Note : Lorsque vous écrivez des variables dans des documents à l'aide de la commande ECRIRE VARIABLES, 4e Dimension utilise un format de données qui lui est propre. Vous ne pouvez récupérer les variables qu'avec la commande LIRE VARIABLES. N'utilisez pas les commandes RECEVOIR VARIABLE ou RECEVOIR PAQUET pour lire un document créé par ECRIRE VARIABLES.

ATTENTION : La commande ECRIRE VARIABLES ne permet pas de sauvegarder les variables de type Tableau. Pour cela, vous devez utiliser les nouvelles commandes du thème BLOB.

Exemples

L'exemple suivant enregistre trois variables dans un fichier nommé PrefsUti :

```
⇒ ECRIRE VARIABLES ("PrefsUti"; VSNom; VLCode; VGIconPict)
```

Variables et ensembles système

Si l'opération s'est correctement déroulée, la variable OK prend la valeur 1, sinon elle prend la valeur 0.

Référence

BLOB VERS DOCUMENT, BLOB VERS VARIABLE, DOCUMENT VERS BLOB, LIRE VARIABLES, VARIABLE VERS BLOB.

LIRE VARIABLES (doc; variable{; variable2; ...; variableN})

Paramètre	Type		Description
doc	Alpha	→	Document contenant la ou les variable(s) à lire
variable	Variable	→	Nom de(s) variable(s) devant recevoir les valeurs

Description

La commande LIRE VARIABLES charge une ou plusieurs variables depuis le document désigné par doc. Ce document doit avoir été créé à l'aide de la commande ECRIRE VARIABLES.

Les variables variable, variable2... variableN sont soit créées, soit réécrites si elles existent déjà.

Si vous passez une chaîne vide ("") dans doc, une boîte de dialogue standard d'ouverture de fichiers apparaît, permettant à l'utilisateur de sélectionner le document à ouvrir. Dans ce cas, la variable système Document contiendra le nom du document choisi.

Dans le cadre de bases compilées, les variables utilisées doivent être du même type que celles chargées du disque.

ATTENTION : Cette commande ne traite pas les variables de type Tableau. Pour cela, vous devez utiliser les nouvelles commandes du thème BLOB.

Exemple

L'exemple suivant charge trois variables d'un document nommé PrefsUti :

```
⇒ LIRE VARIABLES ( "PrefsUti"; VSNom; VLCode; VGIconPict)
```

Variables et ensembles système

La variable système OK prend la valeur 1 si les variables ont été correctement chargées, sinon elle prend la valeur 0.

Référence

BLOB VERS DOCUMENT, BLOB VERS VARIABLE, DOCUMENT VERS BLOB, RECEVOIR VARIABLE, VARIABLE VERS BLOB.

EFFACER VARIABLE (variable)

Paramètre	Type		Description
variable	Variable	→	Nom de la variable à effacer

Description

La commande EFFACER VARIABLE fonctionne différemment en mode interprété et en mode compilé.

Mode interprété
EFFACER VARIABLE efface variable de la mémoire. En conséquence, la variable devient indéfinie : tenter de lire sa valeur génère une erreur de syntaxe. A noter que si par la suite vous assignez de nouveau une valeur à la variable, 4D, en mode interprété, la recréera à la volée. Lorsqu'une variable est effacée, la fonction Indefinie, si elle lui est appliquée, retourne Vrai.

Mode compilé
EFFACER VARIABLE réinitialise uniquement variable à la valeur par défaut de son type (par exemple chaîne vide pour les types Alpha et Texte, 0 — zéro — pour les variables numériques, aucun élément pour un tableau etc.). La variable existe toujours : les variables ne sont jamais indéfinies dans du code compilé.

La variable passée dans variable doit être une variable process ou interprocess.

Note : Il n'est pas nécessaire d'effacer les variables process à la fin de l'exécution d'un process, 4D s'en charge automatiquement.

Les variables locales, c'est-à-dire celles dont le nom est précédé du symbole dollar ($), ne peuvent être effacées par EFFACER VARIABLE. Toutefois, chaque variable locale est automatiquement effacée à la fin de l'exécution de la méthode dans laquelle elle a été créée.

Exemple

Dans un formulaire, vous utilisez une liste déroulante appelée asMalListeD n'ayant qu'un rôle d'interface utilisateur. Autrement dit, vous exploitez ce tableau lors de la saisie de données, mais une fois que le formulaire est refermé, vous n'en avez plus besoin. Par conséquent, lors de l'événement Sur libération, vous effacez simplement le tableau :

```
    ` Méthode objet liste déroulante asMalListeD
Au cas ou
    : (Evenement formulaire=Sur chargement)
        ` Initialiser le tableau comme vous le souhaitez...
        TABLEAU ALPHA(63;asMalListeD;...)
        ` ...
    :(Evenement formulaire=Sur libération)
        ` Vous n'avez plus besoin du tableau
⇒       EFFACER VARIABLE (asMalListeD)
        ` ...
Fin de cas
```

Référence

Indefinie.

Indefinie (variable) → Booléen

Paramètre	Type		Description
variable	Variable	→	Variable à tester
Résultat	Booléen	←	Vrai = Variable actuellement indéfinie Faux = Variable actuellement définie

Description

Indefinie retourne Vrai si variable n'a pas été définie, et Faux si variable a été définie. Une variable est définie si une valeur lui a été assignée. Une variable est indéfinie si aucune valeur ne lui a été assignée, ou si elle a été effacée par la commande EFFACER VARIABLE.

Si la base de données a été compilée avec 4D Compiler, la fonction Indefinie retourne Faux pour toutes les variables.

Exemples

(1) Avant la version 6 de 4e Dimension, la fonction Indefinie fournissait un moyen sûr de tester si la base était exécutée en mode interprété ou en mode compilé :

```
    touteVar:="Hello"
    EFFACER VARIABLE(touteVar)
⇒   Si (Indefinie(touteVar))
        ` Vous êtes en mode interprété
    Sinon
        ` Vous êtes en mode compilé
    Fin de si
```

A compter de la version 6, il est plus pratique d'utiliser la fonction intégrée Application compilee.

(2) Votre application gère un process lorsqu'une commande de menu d'un module particulier de la base est sélectionnée : si le process est déjà créé, vous le passez au premier plan ; s'il n'est pas créé, vous le démarrez. Pour cela, pour chaque module de votre application, vous gérez une variable interprocess ◊PID_... initialisée dans la méthode base Sur ouverture.

Au cours du développement de la base, vous ajoutez de nouveaux modules. Au lieu de devoir à chaque fois modifier la méthode base Sur ouverture (pour ajouter l'initialisation de la variable ◊PID_... correspondante) puis quitter et réouvrir la base pour tout réinitialiser, vous utilisez la fonction Indefinie pour gérer "à la volée" l'ajout d'un nouveau module :

```
      ` Méthode projet M_AJOUT_CLIENTS

      ` Prise en compte des étapes de développement intermédiaires
⇒     Si (Indefinie(◊PID_AJOUT_CLIENTS))
         C_ENTIER LONG(◊PID_AJOUT_CLIENTS)
         ◊PID_AJOUT_CLIENTS:=0
      Fin de si

      Si (◊PID_AJOUT_CLIENTS=0)
         ◊PID_AJOUT_CLIENTS:=Nouveau process("P_AJOUT_CLIENTS";64*1024;
                                                "P_AJOUT_CLIENTS")
      Sinon
         MONTRER PROCESS(◊PID_AJOUT_CLIENTS)
         PASSER AU PREMIER PLAN(◊PID_AJOUT_CLIENTS)
      Fin de si
         ` Note: P_AJOUT_CLIENTS, la méthode de gestion des process,
         ` fixe ◊PID_ADD_CUSTOMERS à zéro lorsqu'elle est terminée.
```

Référence

EFFACER VARIABLE.

57

Codes d'erreurs

Le tableau suivant liste les codes et les messages des erreurs de syntaxe qui peuvent survenir lors de l'exécution de votre code en mode Utilisation ou Menus créés. Quelques erreurs peuvent se produire en mode interprété seulement, quelques-unes en mode compilé seulement et les autres dans les deux modes. Ces erreurs peuvent être interceptées par une méthode d'appel sur erreur installée par la commande APPELER SUR ERREUR.

Code	Description
1	Il manque une parenthèse ouvrante.
2	Il manque un champ.
3	Cette fonction ne peut être appliquée que sur un champ appartenant à une sous-table.
4	Les arguments de la liste doivent tous être du même type.
5	Impossible de déterminer sur quelle table appliquer cette fonction
6	Cette fonction ne peut être exécutée que sur un champ de type sous-table.
7	Il manque un argument de type numérique.
8	Il manque un argument de type alphanumérique.
9	Il manque le résultat d'une condition.
10	Cette fonction ne peut être appliquée à ce type de données.
11	Cette fonction ne peut être appliquée entre deux conditions.
12	Cette fonction ne peut être appliquée entre deux arguments numériques.
13	Cette fonction ne peut être appliquée entre deux arguments alphanumériques.
14	Cette fonction ne peut être appliquée entre deux arguments de type date.
15	Les arguments de cette opération ne sont pas compatibles.
16	Ce champ ne possède pas de lien.
17	Il manque une table.
18	Les types sont incompatibles.
19	Le champ n'est pas indexé.
20	Il manque le signe égal (=).
21	Cette méthode n'existe pas.
22	Les champs doivent appartenir à la même table (ou à la même sous-table) pour un tri ou un graphe.
23	Il manque le signe inférieur (<) ou supérieur (>).
24	Il manque un point-virgule (;).
25	Il y a trop de champs pour le tri.
26	Le champ ne doit pas être de type image, texte, BLOB ou sous-table.
27	Le nom du champ doit être préfixé par le nom de la table auquel il appartient.
28	Le champ doit être du type numérique.

29	La valeur doit être égale à 1 ou 0.
30	Il manque une variable.
31	Aucune barre de menus ne porte ce numéro.
32	Il manque une date.
33	Méthode ou fonction non implémentée
34	Les fichiers comptables ne sont pas ouverts.
35	La table et l'ensemble ne sont pas associés.
36	Nom de table incorrect
37	Il manque le signe d'affectation (:=).
38	Ceci est une fonction et non une méthode.
39	Cet ensemble n'existe pas.
40	Ceci est une méthode et non une fonction.
41	Il manque une variable ou un sous-champ.
42	L'enregistrement ne peut pas être dépilé.
43	La fonction est introuvable.
44	La méthode est introuvable.
45	Il manque une variable ou un champ.
46	Il manque un argument de type alphanumérique ou numérique.
47	Le champ doit être de type alphanumérique.
48	Erreur de syntaxe
49	Impossible d'utiliser cet opérateur ici
50	Ces opérateurs ne peuvent pas être utilisés conjointement.
51	Ce module n'est pas implémenté.
52	Il manque un argument de type tableau.
53	L'indice du tableau est en dehors des limites.
54	Les arguments sont incompatibles.
55	Il manque un argument de type booléen.
56	Il manque un champ, une variable ou une table.
57	Il manque un opérateur.
58	Il manque une parenthèse fermante.
59	Type d'argument inattendu
60	Impossible de passer un paramètre ou une variable locale à une commande EXECUTER sur une base compilée
61	Impossible de modifier le type d'un tableau dans une base compilée
62	Impossible d'appliquer cette commande à une sous-table
63	Le champ n'est pas indexé.
64	Il manque un champ ou une variable de type image.
65	La valeur doit comporter 4 caractères.
66	La valeur doit être composée d'au plus 3 caractères.
67	Cette commande ne peut pas être exécutée sur 4D Server.
68	Il manque une liste.

Astuces

Certains codes d'erreurs signalent des erreurs de syntaxe dûes à des fautes de frappe. Par exemple, vous obtenez l'erreur 37 ("Il manque le signe d'affectation (:=).") si vous exécutez l'expression v=0 alors que vous vouliez écrire v:=0. Dans ce cas, vous éliminez l'erreur en corrigeant votre code dans l'éditeur de méthodes.

Certains codes d'erreurs signalent de simples erreurs de programmation. Par exemple, vous obtenez l'erreur 5 ("Impossible de déterminer sur quelle table appliquer cette fonction.") si vous avez exécuté une commande telle que AJOUTER ENREGISTREMENT sans indiquer de nom de table dans le paramètre correspondant, et vous n'avez pas défini de table par défaut à l'aide de la commande TABLE PAR DEFAUT. Dans ce cas, vous corrigez l'erreur en définissant une table par défaut ou en passant un nom de table dans le paramètre correspondant.

Certains codes d'erreurs signalent des erreurs liées à la structure de la base. Par exemple, vous obtenez l'erreur 16 ("Ce champ ne possède pas de lien.") si vous appliquez la commande CHARGER SUR LIEN à un champ qui n'est pas lié à un autre champ. Dans ce cas, vous éliminez l'erreur en modifiant votre code ou en créant un lien à partir du champ.

Certaines erreurs qui surviennent ne stoppent pas toujours l'exécution de votre code au "bon" endroit. Par exemple, si dans une sous-routine vous recevez l'erreur 53 ("L'indice du tableau est en dehors des limites.") sur la ligne vpChamp:=Champ($1;$2), l'erreur est dûe à des numéros incorrects de table ou de champ passés à la sous-routine en tant que paramètres. Donc, l'erreur se trouve dans la méthode appelante et non à l'endroit où l'erreur est détectée. Dans ce cas, tracez votre code dans la fenêtre de déboguage et recherchez la ligne qui contient l'erreur et puis corrigez-la dans l'éditeur des méthodes.

Référence

APPELER SUR ERREUR.

Le tableau suivant liste les codes d'erreurs générées par le moteur de base de données de 4e Dimension. Ces erreurs de bas niveau peuvent se produire lors d'opérations liées au moteur telles que des interruptions utilisateur, des erreurs de privilèges ou des objets endommagés.

Code	Description
1006	Interruption générée par l'utilisateur.
-9937	Le système de mots de passe est verrouillé par un autre utilisateur.
-9938	L'enregistrement courant a été modifié depuis le trigger.
-9939	Routine externe introuvable.
-9940	L'initialisation de l'extension 4D a échoué.
-9941	Sélecteur EX_GESTALT inconnu.
-9942	Licence 4D Client incompatible avec cette version de 4D Server.
-9943	Erreur de version de plug-in de connectivité 4D.
-9944	Cet utilisateur n'appartient pas au groupe d'accès par 4D Open.
-9945	Erreur 4D Runtime CD-ROM, l'écriture de données est impossible.
-9946	Impossible d'effacer cette sélection temporaire car elle n'existe pas.
-9947	L'option “Autoriser les connexions 4D Open” n'est pas sélectionnée.
-9948	Une fenêtre modale est active.
-9949	Erreur de licence ou de privilège.
-9950	Le numéro d'ordre de ce segment de données n'est pas le bon.
-9951	Ce champ ne possède pas de lien.
-9952	Mauvais en-tête du segment principal.
-9953	Il n'y a pas de fichier d'historique.
-9954	Aucun enregistrement courant.
-9955	QuickTime n'est pas installé.
-9956	Les versions de 4D Server et 4D Client sont incompatibles.
-9957	L'énumération est verrouillée.
-9958	Le process ne peut être démarré.
-9959	Le process de sauvegarde est déjà démarré.
-9960	Aucun plug-in de sauvegarde n'est installé.
-9961	Le process de sauvegarde n'est pas démarré.
-9962	Pas de sauvegarde possible car le serveur quitte.
-9963	Numéro d'enregistrement non valide.
-9964	Table de définition de tri incorrecte envoyée par un poste client.
-9965	Table de définition de recherche incorrecte envoyée par un poste client.
-9966	Les types sont incompatibles.
-9967	L'enregistrement ne peut pas être modifié car il ne peut pas être chargé.

-9968 Numéro d'enregistrement hors sélection.
-9969 Type de champ incorrect.
-9970 Le champ n'est pas indexé.
-9971 Le numéro du champ est en-dehors de l'intervalle défini par le poste client.
-9972 Le numéro de la table est en-dehors de l'intervalle défini par le poste client.
-9973 Mauvaise ressource TRIC.
-9974 Cet enregistrement vient d'être détruit.
-9975 Page d'index de transaction non chargeable.
-9976 Cette commande ne peut être exécutée car la base est en cours de sauvegarde.
-9977 Cette sélection n'existe pas.
-9978 Mot de passe incorrect.
-9979 Impossible d'afficher les informations utilisateur.
-9980 Création de table impossible car la structure est verrouillée.
-9981 Table de définition de nom/numéro de champ envoyée par le poste client incorrecte.
-9982 Enregistrement non chargé car hors sélection pour le poste client.
-9983 Attention ! Vous avez installé deux fois le même package de routines externes.
-9984 Détection d'une clé déjà existante lors d'une transaction.
-9985 Détection d'une boucle lors de la suppression
-9986 En attente du déverrouillage d'un enregistrement par le process n°
-9987 D'autres enregistrements sont liés à celui-ci.
-9988 Impossible de charger ce formulaire.
-9989 Structure de la base invalide (la base de données doit être réparée).
-9990 Dépassement du délai en réception.
-9991 Vous n'avez pas l'autorisation d'accès.
-9992 Ce mot de passe existe déjà.
-9993 Barre de menus endommagée (la base de données doit être réparée).
-9994 Communication série interrompue par l'utilisateur. L'utilisateur a appuyé sur les touches Ctrl+Alt+Maj (Windows) ou Commande+Option+Maj (MacOS).
-9995 Limite de la version de démonstration.
-9996 La pile est pleine (trop d'appels récursifs ou en cascade).
-9997 Le nombre maximum d'enregistrements est atteint.
-9998 La clé d'index existe déjà.
-9999 Disque saturé. Impossible de sauvegarder l'enregistrement.
-10500 Adresse de donnée non valide.
-10501 Structure d'index non valide.
-10502 Structure d'enregistrement non valide
-10503 Numéro d'enregistrement non valide
-10504 Numéro de page d'index non valide
-10600 Impossible de lire ce BLOB. Il est peut-être endommagé.

-1 Point d'entrée non valide utilisé par un plug-in
4001 Numéro de table non valide utilisé par un plug-in
4002 Numéro d'enregistrement non valide utilisé par un plug-in

4003 Numéro de champ invalide utilisé par un plug-in
4004 Un plug-in a requis l'enregistrement courant d'une table alors qu'il n'y en a pas

Notes

(1) Bien que certaines de ces erreurs signalent des problèmes sérieux — par exemple (-10502), Structure d'enregistrement non valide — la plupart sont relativement courantes et peuvent peuvent être traitées par une méthode projet APPELER SUR ERREUR. Par exemple, vous intercepterez fréquemment l'erreur –9998, La clé d'index existe déjà si votre application laisse la possibilité de créer des valeurs identiques pour une table qui contient un champ indexé ayant la propriété Unique.

(2) Certaines de ces erreurs ne se produisent jamais au niveau du langage de 4D. Elles ne surviennent et ne peuvent être traitées qu'à un bas niveau par des routines du moteur de la base ou pendant l'utilisation, par exemple, de 4D Backup ou 4D Open.

(3) L'erreur -10503, Numéro d'enregistrement non valide ne signifie pas toujours que la base doit être réparée. Cette erreur peut se produire si vous tentez d'utiliser le numéro (avec par exemple la commande ALLER A ENREGISTREMENT) d'un enregistrement venant d'être créé pendant une transaction. La raison en est que les enregistrements créés lors de transactions reçoivent des numéros temporaires jusqu'à ce que la transaction soit validée. Si l'erreur survient dans ce contexte, votre base est valide, mais pas votre algorithme.

(4) L'erreur -9999 Disque saturé. Impossible de sauvegarder l'enregistrement se produit lorsque tous les segments de votre base sont pleins ou placés sur des volumes pleins. Cette erreur peut également être générée si le fichier de données est verrouillé ou stocké sur un volume verrouillé. Cela vous permet par exemple de détecter, depuis la méthode base Sur ouverture de votre application, des fichiers de données éventuellement verrouillés. Pour plus d'informations sur ce point, reportez-vous à la section Tester le verrouillage du fichier de données.

Référence

APPELER SUR ERREUR.

Le tableau suivant liste les erreurs qui peuvent se produire dans le cadre de l'utilisation des composants réseaux.

Code	Description
-10001	Interruption de la connexion à la base en cours
-10002	Interruption de la connexion pour ce process
-10003	Paramètres de connexion incorrects
-10020	Aucun serveur sélectionné par OP Select 4D server
-10021	Aucun serveur trouvé par OP Find 4D server
-10030	Désynchronisation pendant l'écriture
-10031	Désynchronisation pendant la lecture
-10033	Taille des données incorrecte pendant la lecture
-10050	Option inconnue dans Get/SetOption
-10051	Valeur incorrecte dans Get/SetOption
2	L'utilisateur a cliqué sur le bouton Autres pendant l'utilisation de OP Select 4D Server

Le tableau suivant liste les codes d'erreurs retournés par le gestionnaire de fichiers du système d'exploitation. Ces erreurs peuvent se produire en particulier lorsque vous utilisez les commandes du thème Documents système. Pour plus d'informations, reportez-vous à la section Présentation des documents système.

Code	Description
-33	Le répertoire du disque est plein. Vous ne pouvez pas créer de fichier sur le disque.
-34	Le disque est plein. Il n'y a plus de place disponible sur le disque.
-35	Le volume n'existe pas.
-36	Erreur d'Entrée/Sortie. Il y a probablement un secteur défectueux sur le disque.
-37	Nom de fichier ou de volume incorrect. Il est trop long et/ou comporte un caractère invalide.
-38	Tentative de lecture ou d'écriture dans un fichier non ouvert.
-39	Tentative de lecture après la fin de fichier.
-40	Tentative de lecture ou d'écriture avant le début du fichier.
-41	Mémoire insuffisante pour ouvrir un nouveau fichier.
-42	Trop de fichiers ouverts.
-43	Fichier non trouvé.
-44	Disque physiquement verrouillé.
-45	Fichier verrouillé.
-46	Volume verrouillé par logiciel.
-47	Tentative d'accès à un fichier supprimé.
-48	Tentative d'utilisation du nom d'un fichier déjà supprimé pour renommer un fichier.
-49	Fichier déjà ouvert en Lecture/Ecriture.
-51	Tentative d'accès à un fichier avec un numéro de référence de fichier invalide
-52	Erreur interne du gestionnaire de fichiers (le marqueur de fichiers est perdu).
-53	Le volume a été éjecté.
-54	Tentative d'écriture dans un fichier verrouillé
-57	Tentative d'écriture sur un disque non-Macintosh
-58	Erreur de fichier système
-60	Mauvais secteur de répertoire principal. Votre disque est endommagé.
-61	L'accès en lecture/écriture ne permet pas d'écrire.
-64	Problème physique avec le disque (installation ou formatage incorrect,...)
-84	Problème physique avec le disque (installation ou formatage incorrect,...)
-120	Tentative d'accès à un fichier avec un chemin d'accès spécifiant un répertoire inexistant

-121 Un chemin d'accès ne peut être créé
-124 Tentative d'accès à un volume partagé déconnecté

Référence

APPELER SUR ERREUR.

Le tableau ci-dessous liste les principaux codes d'erreurs retournés par le gestionnaire de mémoire du système d'exploitation.

Code	Description
-108	Mémoire disponible insuffisante. Allouez davantage de mémoire à votre application 4D.
-109	Problème interne de mémoire. La mémoire est probablement endommagée. Quittez l'application dès que possible. Redémarrez la machine et réouvrez la base.
-111	Problème interne de mémoire. La mémoire est probablement endommagée. Quittez l'application dès que possible. Redémarrez la machine et réouvrez la base. (*)
-117	Problème interne de mémoire. La mémoire est probablement endommagée. Quittez l'application dès que possible. Redémarrez la machine et réouvrez la base.

Conseil : Lorsque vous travaillez avec des gros tableaux, des BLOBs, des images ou des ensembles (c'est-à-dire des objets pouvant manipuler de grandes quantités de données), utilisez une méthode projet installée par APPELER SUR ERREUR pour tester l'erreur -108.

(*) Une erreur -111 peut également se produire lorsque vous tentez de lire une valeur dans un BLOB à un offset hors intervalle. Dans ce cas, cette erreur est mineure et vous n'êtes pas obligé de fermer la session de travail. Il vous suffit de corriger l'offset que vous avez passé à la commande BLOB.

Référence

APPELER SUR ERREUR.

Le tableau ci-dessous liste les codes d'erreurs retournés par le gestionnaire d'impression du système d'exploitation. Ces codes peuvent être retournés durant l'impression.

Code	Description
-1	Problème lors de l'enregistrement d'un fichier à imprimer
-27	Problème de communication avec l'imprimante
-128	Impression interrompue par l'utilisateur
-193	Fichier de ressources introuvable
-4100	La connexion avec l'imprimante a été interrompue
-4101	Imprimante éteinte ou non connectée
-8150	Aucune imprimante n'a été sélectionnée
-8151	L'imprimante a été initialisée avec une version du driver différente de la vôtre
-8192	Time out de l'imprimante

Référence

APPELER SUR ERREUR.

Le tableau ci-dessous liste les codes d'erreurs retournés par le gestionnaire de ressources du système d'exploitation.

Code	Description
-1	Impossible d'ouvrir ce fichier de ressources
-192	Ressource introuvable
-193	La ressource est endommagée (le fichier doit être réparé)
-194	La ressource ne peut être ajoutée
-196	La ressource ne peut être supprimée

Référence

APPELER SUR ERREUR.

Le tableau ci-dessous liste les codes NaN retournés par le système d'exploitation. Le sigle NaN signifie “Not a Number”. C'est une représentation du Standard Apple Numeric Environment (SANE) qui est appelée lorsqu'une opération produit un résultat se trouvant dans le domaine de SANE.

Code	Description
1	Racine carrée invalide
2	Addition invalide
4	Division invalide
8	Multiplication invalide
9	Reste invalide
17	Conversion d'une chaîne ASCII invalide
20	Conversion d'un nombre de type Comp en virgule flottante
21	Création d'un NaN avec un code zéro
33	Argument invalide passé à une fonction trig
34	Argument invalide passé à une fonction trig inversée
36	Argument invalide passé à une fonction log
37	Argument invalide passé à une fonction xi ou xy
38	Argument invalide passé à une fonction financière
255	Stockage non initialisé

Le tableau ci-dessous liste les codes retournés par le gestionnaire de son du système d'exploitation.

Code	Description
-203	Trop de commandes de sons
-204	La ressource son ne peut être chargée
-205	Le canal de son est endommagé
-206	Format de ressource son incorrect
-207	Mémoire insuffisante pour jouer le son
-209	Le canal de son est indisponible

Référence

APPELER SUR ERREUR.

Le tableau ci-dessous fournit le code d'erreur retourné par le gestionnaire de port série du système d'exploitation.

Code	Description
-28	Il n'y a pas de port série ouvert

Référence

APPELER SUR ERREUR.

Le tableau ci-dessous liste certaines erreurs courantes retournées par MacOS. Lorsqu'une de ces erreurs se produit, il n'est généralement pas possible de poursuivre la session sans redémarrer.

Code	Description
4	Division par zéro
15	Erreur du chargeur de segments : 4e Dimension n'a pas pu charger un de ses propres segments de code. Vous devez allouer davantage de mémoire à 4e Dimension.
17 à 24	Un élément système est manquant. Vérifiez que votre dossier système a été correctement installé.
25	Mémoire saturée. Vous devez allouer davantage de mémoire à 4e Dimension.
28	La pile a été placée dans la heap de l'application. Vous devez allouer davantage de mémoire à 4e Dimension.

Tester le verrouillage du fichier de données

version 5.2.5

Pour tester le verrouillage du fichier de données et du volume sur lequel il est stocké, vous devez appeler le gestionnaire de fichiers du système d'exploitation. Si vous le faites pour chaque opération de la base qui modifie le fichier de données, les performances du moteur de la base seront considérablement affectées.

Il est de votre ressort de tester le statut du fichier de données au début d'une session de travail. Généralement, vous placez le test dans la méthode base Sur ouverture de la base. Jusqu'à la version 5.2.5 de 4e Dimension et 1.2.5 de 4D Server, tester le verrouillage du fichier de données nécessitait l'utilisation de routines externes qui retournaient les propriétés du fichier de données et du volume sur lequel il était stocké, elles-mêmes transmises par le gestionnaire de fichiers du système.

Pour simplifier le test du statut du fichier de données, 4D et 4D Server signalent maintenant systématiquement un fichier de données verrouillé chaque fois que vous essayez de créer un enregistrement. Si le fichier de données ou son volume sont verrouillés, le moteur de la base génère désormais l'erreur -9999 Disque saturé. Impossible de sauvegarder l'enregistrement.

Le code suivant est un exemple pour tester le verrouillage du fichier de données :

```
   ` Méthode projet Données sont-elles verrouillées
   ` Données sont-elles verrouillées -> Booléen
   ` Données sont-elles verrouillées -> Vrai si fichier verrouillé ou disque plein
gErreur:=0
APPELER SUR ERREUR("Gestion des erreurs")
CREER ENREGISTREMENT([Tout Fichier])
STOCKER ENREGISTREMENT([Tout Fichier])
APPELER SUR ERREUR("")
Si (gErreur=0)
   SUPPRIMER ENREGISTREMENT([Tout Fichier])
Fin de si
$0:=(gErreur=-9999)
```

La méthode installée par APPELER SUR ERREUR, appelée Gestion des erreurs, est listée ci-dessous :

```
   ` Méthode projet Gestion des erreurs
gErreur:=Error
```

Ensuite, dans la méthode base Sur ouverture, vous pouvez écrire :

```
   ` Méthode base Sur ouverture
   ` ...
Si (Données sont-elles verrouillées)
      ` Afficher une boîte de dialogue qui explique que le fichier de données est
      ` verrouillé, ou que le volume sur lequel le fichier de données est stocké est
      ` verrouillé ou plein.
      ` Vous voulez aussi peut-être utiliser la base en mode lecture seulement :
      ` le fichier de données peut se trouver sur un CD-ROM.
      ` Donc, la boîte de dialogue peut comporter un bouton “Utiliser en mode lecture
      ` seulement” ou un bouton “Quitter”.
   Si (bQuitter=1)
         ` Quitter la base et vérifier ce qui se passe au niveau du bureau
      QUITTER 4D
   Sinon
         ` Définir une variable interprocess pour signaler que le fichier de données est
         ` verrouillé
      <>gLECTURE_SEULEMENT:=Vrai
         ` ...
         ` Continuer l'exécution de Sur ouverture
      ` ...
   Fin de si
Fin de si
```

Si vous autorisez l'utilisation de la base avec un fichier de données verrouillé, assurez-vous de restreindre l'accès aux opérations qui peuvent modifier le contenu de la base. Pour ce faire, vous pouvez appeler à nouveau votre fonction de test ou maintenir une variable interprocess comme dans l'exemple ci-dessus. Si, par exemple, votre base comporte une méthode objet M_Ajouter_Clients qui crée un process dans lequel des enregistrements de clients sont créés, vous pouvez savoir à l'avance si cela sera possible ou non. Si ce n'est pas possible, il est inutile de lancer le process. Par exemple :

```
   ` Méthode projet M_Ajouter_Clients
Si (Peut écrire données)
      ` Continuer
   <>vlPREF_CLIENT:=Nouveau process(...;...;...)
      ` ...
Fin de si
```

La méthode Peut écrire données est la suivante :

```
   ` Méthode projet Peut écrire données
   ` Peut écrire données -> Booléen
   ` Peut écrire données -> Vrai si le fichier de données n'est pas verrouillé
$0:=Vrai
Si (<>gLECTURE_SEULEMENT) ` ou Si (Données sont-elles verrouillées)
   ALERTE("Cette opération ne peut pas être effectuée sur un fichier de données en
                                                                    lecture seulement.")
   $0:=Faux
Fin de si
```

Note importante : Pour conserver les performances du moteur de la base de données dans 4D et 4D Server, ne testez pas le statut du fichier de données à chaque opération pouvant modifier le contenu du fichier de données. Si, par exemple, vous ajoutez des enregistrements dans une transaction, souvenez-vous que le fichier de données reste intact jusqu'à ce que vous tentiez de valider la transaction.
Tester le verrouillage du fichier de données lorsque vous effectuez des modifications dans une transaction ajouterait des tests inutiles. Par conséquent, le statut du fichier de données doit être testé seulement lorsque vous essayez de créer un enregistrement hors transaction. Cela comprend l'ajout et l'import de nouveaux enregistrements en mode Utilisation, ainsi que les commandes AJOUTER ENREGISTREMENT, STOCKER ENREGISTREMENT (appliquée à un nouvel enregistrement) et TABLEAU VERS SELECTION (qui crée de nouveaux enregistrements). En revanche, la création de nouveaux enregistrements dans une transaction et la modification ou la suppression d'enregistrements existants ne nécessitent pas de tester le statut du fichier de données.

Note : Tester le statut verrouillé du fichier de données n'empêche pas de continuer à tester les autres erreurs E/S pouvant survenir lors de l'écriture du fichier de données.

Référence

Erreurs de la base de données, Méthode base Sur ouverture.

58

Codes ASCII

Tables des codes ASCII

• La table standard des codes ASCII (de 0 à 127) est identique sur les plates-formes Windows et MacOS.

Les codes ASCII de 0 à 127 sont listés dans les sections Codes ASCII 0..63 et Codes ASCII 64..127.

• La table ASCII étendue (codes ASCII de 128 à 255) est différente entre Windows et MacOS. Afin d'assurer l'indépendance de plate-forme de vos applications, 4e Dimension, lorsque le programme fonctionne sous Windows, convertit automatiquement les codes ASCII (de la table Windows vers la table MacOS) lorsque des caractères sont entrés dans l'environnement 4D (saisie de données, copier/coller, import d'enregistrements, etc.) ou encore (de la table MacOS vers la table Windows) lorsque des caractères sont extraits de l'environnement 4D (couper ou copier, export, etc.).

Les codes ASCII de 127 à 255 sont listés dans les sections Codes ASCII 128..191 et Codes ASCII 192..255.

4e Dimension et les codes ASCII

Sur les deux plates-formes MacOS et Windows, le moteur interne de base de données et le langage de de 4D travaillent avec la table ASCII étendue du Macintosh. Lorsque vous saisissez des données (ajout d'enregistrements, édition de méthodes, etc.), 4e Dimension utilise le schéma interne de conversion d'Altura pour convertir les codes provenant du clavier (qui sont donc exprimés à l'aide de la table ASCII étendue Windows) en codes Macintosh. Par exemple, pour saisir le caractère “ß”, vous tapez **Alt**+0223, mais c'est le code ASCII 167 que 4e Dimension va stocker dans l'enregistrement. Ce mode de fonctionnement est totalement transparent pour l'utilisateur car lorsque vous effectuez par exemple une recherche, vous saisissez la valeur réelle à trouver dans l'éditeur de recherches. La valeur que vous tapez (**Alt**+0223) est également convertie en code ASCII 167, et la recherche aboutira.

Le même principe est appliqué lorsque vous tapez **Alt**+0223 dans l'éditeur de méthodes. Notez cependant que si vous recherchez un caractère sur la base de son code ASCII, vous devrez utiliser le code ASCII Macintosh du caractère.

Par exemple :

CHERCHER (...; [MaTable]MonChamp="ß") ` ß s'obtient par Alt+0223

est identique à :

CHERCHER (...;[MaTable]MonChamp=**Caractere**(167))
` ß a pour code ASCII MacOS 167

Référence

APPELER SUR EVENEMENT, Code ascii, ISO vers Mac, Mac vers ISO, Mac vers Windows, Windows vers Mac.

La table standard des codes ASCII (de 0 à 127) est commune aux plates-formes Windows et Macintosh.

Car	Dec	Hex	Car	Dec	Hex
Nul	0	0	SP	32	20
SOH	1	1	!	33	21
STX	2	2	"	34	22
ETX	3	3	#	35	23
EOT	4	4	$	36	24
ENQ	5	5	%	37	25
ACK	6	6	&	38	26
BEL	7	7	'	39	27
BS	8	8	(	40	28
HT	9	9	)	41	29
LF	10	A	*	42	2A
VT	11	B	+	43	2B
FF	12	C	,	44	2C
CR	13	D	-	45	2D
SO	14	E	.	46	2E

SI	15	F	/	47	2F
DLE	16	10	0	48	30
DC1	17	11	1	49	31
DC2	18	12	2	50	32
DC3	19	13	3	51	33
DC4	20	14	4	52	34
NAK	21	15	5	53	35
SYN	22	16	6	54	36
ETB	23	17	7	55	37
CAN	24	18	8	56	38
EM	25	19	9	57	39
SUB	26	1A	:	58	3A
ESC	27	1B	;	59	3B
FS	28	1C	<	60	3C
GS	29	1D	=	61	3D
RS	30	1E	>	62	3E
US	31	1F	?	63	3F

Référence

APPELER SUR EVENEMENT, Caractere, ISO vers Mac, Mac vers ISO, Mac vers Windows, Windows vers Mac.

La table standard des codes ASCII (de 0 à 127) est commune aux plates-formes Windows et Macintosh.

Car	Dec	Hex	Car	Dec	Hex
@	64	40	`	96	60
A	65	41	a	97	61
B	66	42	b	98	62
C	67	43	c	99	63
D	68	44	d	100	64
E	69	45	e	101	65
F	70	46	f	102	66
G	71	47	g	103	67
H	72	48	h	104	68
I	73	49	i	105	69
J	74	4A	j	106	6A
K	75	4B	k	107	6B
L	76	4C	l	108	6C
M	77	4D	m	109	6D

N	78	4E	n	110	6E
O	79	4F	o	111	6F
P	80	50	p	112	70
Q	81	51	q	113	71
R	82	52	r	114	72
S	83	53	s	115	73
T	84	54	t	116	74
U	85	55	u	117	75
V	86	56	v	118	76
W	87	57	w	119	77
X	88	58	x	120	78
Y	89	59	y	121	79
Z	90	5A	z	122	7A
[	91	5B	{	123	7B
\	92	5C	|	124	7C
]	93	5D	}	125	7D
^	94	5E	~	126	7E
_	95	5F	Del	127	7F

Référence

APPELER SUR EVENEMENT, Code ascii, ISO vers Mac, Mac vers ISO, Mac vers Windows, Windows vers Mac.

Le tableau suivant présente les caractères affichés par 4e Dimension pour les codes ASCII de la table étendue. Cette table s'applique aux versions Macintosh et Windows de 4D. De plus, ce tableau indique les combinaisons de touches à appliquer pour afficher les caractères sous Windows (**Alt**+code ASCII Windows).

Table ASCII étendue Macintosh (traitement interne 4D)			Correspondance codes ASCII étendus Windows	
Dec	Hex	Caractère (Times)	Combinaison Alt +	Caractère (Arial)
128	80	Ä	0196	Ä
129	81	Å	0197	Å
130	82	Ç	0199	Ç
131	83	É	0201	É
132	84	Ñ	0209	Ñ
133	85	Ö	0214	Ö
134	86	Ü	0220	Ü
135	87	á	0225	á
136	88	à	0224	à
137	89	â	0226	â
138	8A	ä	0228	ä
139	8B	ã	0227	ã
140	8C	å	0229	å
141	8D	ç	0231	ç
142	8E	é	0233	é

143	8F	è	0232	è
144	90	ê	0234	ê
145	91	ë	0235	ë
146	92	í	0237	í
147	93	ì	0236	ì
148	94	î	0238	î
149	95	ï	0239	ï
150	96	ñ	0241	ñ
151	97	ó	0243	ó
152	98	ò	0242	ò
153	99	ô	0244	ô
154	9A	ö	0246	ö
155	9B	õ	0245	õ
156	9C	ú	0250	ú
157	9D	ù	0249	ù
158	9E	û	0251	û
159	9F	ü	0252	ü

160	A0	†	0134	†
161	A1	°	0176	°
162	A2	¢	0162	¢
163	A3	£	0163	£
164	A4	§	0167	§
165	A5	•	0149	•
166	A6	¶	0182	¶
167	A7	ß	0223	ß
168	A8	®	0174	®
169	A9	©	0169	©
170	AA	™	0153	™
171	AB	´	0145	´
172	AC	¨	0168	¨
173	AD	≠		
174	AE	Æ	0198	Æ
175	AF	Ø	0216	Ø

176	B0	∞		
177	B1	±	0177	±
178	B2	≤		
179	B3	≥		
180	B4	¥	0165	¥
181	B5	µ	0181	µ
182	B6	∂		
183	B7	∑		
184	B8	∏		
185	B9	π		
186	BA	∫		
187	BB	ª	0170	ª
188	BC	º	0186	º
189	BD	Ω		
190	BE	æ	0230	æ
191	BF	ø	0248	ø

Note : Les cases grisées signalent des caractères non disponibles sous Windows, ou différents des caractères Macintosh.

Référence

APPELER SUR EVENEMENT, Caractere, ISO vers Mac, Mac vers ISO, Mac vers Windows, Windows vers Mac.

Le tableau suivant présente les caractères affichés par 4e Dimension pour les codes ASCII de la table étendue. Cette table s'applique aux versions Macintosh et Windows de 4D. De plus, ce tableau indique les combinaisons de touches à appliquer pour afficher les caractères sous Windows (**Alt**+code ASCII Windows).

Table ASCII étendue Macintosh (traitement interne 4D)			Correspondance codes ASCII étendus Windows	
Dec	**Hex**	**Caractère (Times)**	**Combinaison Alt +**	**Caractère (Arial)**
192	C0	¿	0191	¿
193	C1	¡	0161	¡
194	C2	¬	0172	¬
195	C3	√		
196	C4	ƒ	0131	ƒ
197	C5	≈		
198	C6	Δ		
199	C7	«	0171	«
200	C8	»	0187	»
201	C9	…	0133	…
202	CA	(espace)	0160	(espace)
203	CB	À	0192	À
204	CC	Ã	0195	Ã
205	CD	Õ	0213	Õ
206	CE	Œ	0140	Œ

207	CF	œ	0156	œ
208	D0	–	0150	–
209	D1	—	0151	—
210	D2	“	0147	“
211	D3	”	0148	”
212	D4	‘	0145	‘
213	D5	’	0146	’
214	D6	÷	0247	÷
215	D7	◊		
216	D8	ÿ	0255	ÿ
217	D9	Ÿ	0159	Ÿ
218	DA	⁄		
219	DB	¤	0164	¤
220	DC	‹	0139	‹
221	DD	›	0155	›
222	DE	fi		
223	DF	fl		

224	E0	‡	0135	‡
225	E1	·	0183	·
226	E2	‚		
227	E3	„	0132	„
228	E4	‰	0137	‰
229	E5	Â	0194	Â
230	E6	Ê	0202	Ê
231	E7	Á	0193	Á
232	E8	Ë	0203	Ë
233	E9	È	0200	È
234	EA	Í	0205	Í
235	EB	Î	0206	Î
236	EC	Ï	0207	Ï
237	ED	Ì	0204	Ì
238	EE	Ó	0211	Ó
239	EF	Ô	0212	Ô

240	F0			
241	F1	Ò	0210	Ò
242	F2	Ú	0218	Ú
243	F3	Û	0219	Û
244	F4	Ù	0217	Ù
245	F5	ı	0185	ı
246	F6	ˆ		
247	F7	˜	0152	~
248	F8	¯	0175	¯
249	F9	˘		
250	FA	˙		
251	FB	˚		
252	FC	¸	0184	¸
253	FD	˝		
254	FE	˛		
255	FF	ˇ		

Note : Les cases grisées signalent des caractères non disponibles sous Windows, ou différents des caractères Macintosh.

Référence

APPELER SUR EVENEMENT, Code ascii, ISO vers Mac, Mac vers ISO, Mac vers Windows, Windows vers Mac.

Codes des touches de fonction

4e Dimension retourne le code clavier des touches de fonction dans la variable système Keycode, qui est utilisée dans les méthodes projet installées par la commande APPELER SUR EVENEMENT pour intercepter les événements.

Les valeurs des touches de fonctions ne sont pas basées sur des codes ASCII. Elles sont listées ci-dessous :

Touche	Code
F1	-122
F2	-120
F3	-99
F4	-118
F5	-96
F6	-97
F7	-98
F8	-100
F9	-101
F10	-109
F11	-103
F12	-111
F13	-105
F14	-107
F15	-113

Par ailleurs, le tableau suivant liste les valeurs retournées dans la variable système Keycode lorsque vous appuyez sur les touches standard comme Retour chariot ou Entrée.

Touche	Code
Entrée	3
Retour chariot	13
Retour arrière	8
Tabulation	9
Echappement	27
Effacement	127
Aide	5
Début	1
Fin	4
Haut de page	11
Bas de page	12
Flèche gauche	28
Flèche droite	29
Flèche haut	30
Flèche bas	31

Référence

APPELER SUR EVENEMENT.

59

Syntaxe des commandes

A

Abs (nombre) → Numérique
Activation → Booléen
ACTIVER BOUTON ({*; }objet)
ACTIVER LIGNE MENU (menu; ligneMenu{; process})
ADJOINDRE ELEMENT ({table; }ensemble)
AFFICHER BARRE DE MENUS
AFFICHER BARRE OUTILS
AFFICHER ENREGISTREMENT {(table)}
AFFICHER FENETRE {(fenêtre)}
Ajouter a date (date; années; mois; jours) → Date
Ajouter a document (document{; type}) → docRéf
AJOUTER A LISTE (liste; texteElément; numElément{; sous-liste; déployée})
AJOUTER A PRESSE PAPIERS (typeDonnées; données)
AJOUTER ENREGISTREMENT ({table}{; }{*})
AJOUTER LIGNE MENU (menu; libelléLigne{; process})
AJOUTER SEGMENT DE DONNEES
AJOUTER SOUS ENREGISTREMENT (sousTable; formulaire{; *})
ALERTE (message{; libelléBoutonOK})
ALLER A CHAMP ({*; }objet)
ALLER A DERNIER ENREGISTREMENT {(table)}
ALLER A DERNIER SOUS ENREGISTREMENT (sousTable)
ALLER A ENREGISTREMENT ({table; }enregistrement)
ALLER A PAGE (numéroPage)
ALLER DANS SELECTION ({table; }position)
Ancien (champ) → Expression
ANCIEN LIEN RETOUR (champ)
Annee de (date) → Numérique
ANNULER TRANSACTION
Appartient a ensemble (ensemble) → Booléen
Appartient au groupe (utilisateur; groupe) → Booléen
Appel exterieur → Booléen
APPELER 4D
APPELER PROCESS (process)
APPELER SUR A PROPOS (libelléLigne; méthode)
APPELER SUR ERREUR (méthodErreur)
APPELER SUR EVENEMENT (méthodeEvén{; nomProcess})
Application compilee → Booléen
APPLIQUER A SELECTION ({table; }formule)
APPLIQUER A SOUS SELECTION (sousTable; formule)

Apres → Booléen
Arctan (nombre) → Numérique
ARRETER SERVEUR WEB
Arrondi (nombre; nbDécimales) → Numérique
ASSOCIER TYPES FICHIER (macOS; windows; contexte)
Avant → Booléen
Avant selection {(table)} → Booléen
Avant sous enregistrement (sousTable) → Booléen

B

BEEP
BLOB VERS DOCUMENT (document; blob{; *})
BLOB vers entier (blob; ordreOctet{; offset}) → Numérique
BLOB vers entier long (blob; ordreOctet{; offset}) → Numérique
BLOB VERS IMAGE (blobImage; image)
BLOB vers liste (blob{; offset}) → RéfListe
BLOB vers reel (blob; formatRéel{; offset}) → Numérique
BLOB vers texte (blob; formatTexte{; offset{; longueurTexte}}) → Chaîne
BLOB VERS VARIABLE (blob; variable{; offset})

C

CACHER BARRE DE MENUS
CACHER BARRE OUTILS
CACHER FENETRE {(fenêtre)}
CACHER PROCESS (process)
Caractere (codeASCII) → Chaîne
Chaine (expression{; format}) → Alpha
Chaine heure (secondes) → Alpha
Champ (tableNum | champPtr{; champNum}) → Num | Pointeur
CHANGER BARRE (barreNum{; process}{; *})
CHANGER COORDONNEES FENETRE (gauche; haut; droite; bas{; fenêtre})
CHANGER CREATEUR DOCUMENT (document; créateur)
CHANGER ELEMENT (liste; réfElément; textElément; nouvelRéf{; sous-liste; déployée})
CHANGER JEU DE CARACTERES ({*; }objet; police)
CHANGER LICENCES
CHANGER MOT DE PASSE (motDePasse)
CHANGER POINTEUR SOURIS {(curseur)}
CHANGER POSITION DANS DOCUMENT (docRef; offset{; ancre})
CHANGER PRIVILEGES
CHANGER PROPRIETES DOCUMENT (document; verrouillé; invisible; créé le; créé à; modifié le; modifié à)
CHANGER PROPRIETES ELEMENT (liste; réfElément; saisissable; styles; icône)
CHANGER PROPRIETES LISTE (liste; apparence{; icône{; ligneHauteur{; doubleClic}}})
CHANGER RACCOURCI CLAVIER (menu; cmdeMenu; touche{; process})

CHANGER STYLE ({*; }objet; style)
CHANGER STYLE LIGNE MENU (menu; ligneMenu; styleLigne{; process})
CHANGER TAILLE ({*; }objet; taille)
CHANGER TAILLE DOCUMENT (document; taille)
CHANGER TEXTE LIGNE MENU (menu; ligneMenu; texteLigne{; process})
CHANGER TITRE FENETRE (titre{; fenêtre})
CHANGER TYPE DOCUMENT (document; type)
CHANGER UTILISATEUR
CHARGER ANCIEN (champ)
CHARGER ENREGISTREMENT {(table)}
CHARGER ENSEMBLE ({table; }ensemble; document)
CHARGER ET COMPRESSER IMAGE (document; type; qualité; image)
Charger liste (nomListe) → RéfListe
CHARGER SUR LIEN (tableN | champN{; discriminant})
CHERCHER ({table}{; critèreRecherche}{; *})
CHERCHER DANS SELECTION ({table}{; critère}{; *})
Chercher dans tableau (tableau; valeur{; début}) → Numérique
Chercher fenetre (gauche; haut{; partieFenêtre}) → RefFenêtre
CHERCHER PAR EXEMPLE ({table}{; }{* })
CHERCHER PAR FORMULE ({table}{; }{formule})
CHERCHER PAR FORMULE DANS SELECTION ({table}{; }{formule})
CHERCHER PAR TABLEAU (champIndexé; tableau)
Chercher process (nom{; *}) → Numérique
CHERCHER SOUS ENREGISTREMENTS (sousTable; formule)
CHERCHER SUR CLE
CHOIX COULEUR ({*; }objet; couleur{; arrièrePlan})
CHOIX ENUMERATION ({*; }objet; énum)
CHOIX FILTRE SAISIE ({*; }objet; filtreSaisie)
CHOIX FORMATAGE ({*; }objet; formatAffich)
CHOIX SAISISSABLE ({*; }objet; zoneSaisie)
CHOIX VISIBLE ({*; }objet; visible)
Code ascii (caractère) → Numérique
COMPRESSER BLOB (blob{; compression})
COMPRESSER FICHIER IMAGE (document; type; qualité)
COMPRESSER IMAGE (image; type; qualité)
CONFIRMER (message{; libelléBoutonOK{; libelléBoutonAnn}})
Connexion Web securisee → Booléen
Contexte Web → Booléen
Convertisseur Euro (valeur; deMonnaie; versMonnaie) → Numérique
COORDONNEES ECRAN (gauche; haut; droite; bas{; écran})
COORDONNEES FENETRE (gauche; haut; droite; bas{; fenêtre})
COPIER BLOB (srcBLOB; dstBLOB; srcOffset; dstOffset; nombre)
COPIER DOCUMENT (nomSource; nomDest{; *})
COPIER ENSEMBLE (srcEns; dstEns)
Copier liste (liste) → RéfListe
COPIER SELECTION ({table; }tempo)
COPIER TABLEAU (source; destination)

Cos (nombre) → Numérique
Createur document (document) → Alpha
CREER ALIAS (cheminCible; cheminAlias)
Creer document (document{; type}) → docRéf
CREER DOSSIER (cheminAccès)
CREER ENREGISTREMENT {(table)}
CREER ENSEMBLE SUR TABLEAU (table; tabEnrg; {nomEnsemble})
Creer fenetre (gauche; haut; droite; bas{;type{;titre{;caseFermeture}}}){ → RefFen }
Creer fenetre externe (gauche; haut; droit; bas; type; titre; zonePlugin) → Numérique
Creer fenetre formulaire ({table;} nomForm{; type{; posH{; posV{; *}}}}) → Entier long
Creer fichier ressources (resNomFichier{; typeFichier}) → DocRef
CREER IMAGETTE (source; dest{; largeur{; hauteur{; mode{; profondeur}}}})
CREER SELECTION SUR TABLEAU (table; tabEnrg {; nomSélection})
CREER SOUS ENREGISTREMENT (sousTable)
CREER SUR LIEN (champ)
CRYPTER BLOB (aCrypter; cléPrivEmetteur{; cléPubRécepteur})
CUMULER SUR (objet{; objet2; ...; objetN})
C_ALPHA ({méthode; }taille; variable{; variable2; ...; variableN})
C_BLOB ({méthode; }variable{; variable2; ...; variableN})
C_BOOLEEN ({méthode; }variable{; variable2; ...; variableN})
C_DATE ({méthode; }variable{; variable2; ...; variableN})
C_ENTIER ({méthode; }variable{; variable2; ...; variableN})
C_ENTIER LONG ({méthode; }variable{; variable2; ...; variableN})
C_GRAPHE ({méthode; }variable{; variable2; ...; variableN})
C_HEURE ({méthode; }variable{; variable2; ...; variableN})
C_IMAGE ({méthode; }variable{; variable2; ...; variableN})
C_POINTEUR ({méthode; }variable{; variable2; ...; variableN})
C_REEL ({méthode; }variable{; variable2; ...; variableN})
C_TEXTE ({méthode; }variable{; variable2; ...; variableN})

D

Date (chaîneDate) → Date
Date du jour {(*)} → Date
DEBUT SELECTION {(table)}
DEBUT SOUS ENREGISTREMENT (sousTable)
DEBUT TRANSACTION
Dec (nombre) → Numérique
DECOMPRESSER BLOB (blob)
DECRYPTER BLOB (aDécrypter; cléPubEmetteur{; cléPrivRécepteur})
Demander (message{; réponseDéfaut{; titreBoutonOK{; titreBoutonAnn}}}) → Alpha
DEPILER ENREGISTREMENT {(table)}
DEPLACER DOCUMENT (cheminSource; cheminDest)
DEPLACER FENETRE
DEPLACER OBJET ({*; } objet; dépH; dépV{; redimH{; redimV}}{; *})
DEPLACER SELECTION ({table; }tempo)

Dernier objet → Pointeur
DERNIERE PAGE
Desactivation → Booléen
DESINSCRIRE CLIENT
DIALOGUE ({table; }formulaire)
DIFFERENCE (ensemble1; ensemble2; résultat)
DOCUMENT VERS BLOB (document; blob{; *})
Dossier 4D → Alpha
Dossier systeme ({type}) → Alpha
Dossier temporaire → Alpha
DUPLIQUER ENREGISTREMENT {(table)}

E

Ecart type (séries) → Numérique
Ecran principal → Entier long
ECRIRE CACHE
ECRIRE FICHIER IMAGE (nomFichier; image{; format})
ECRIRE IMAGE DANS BIBLIOTHEQUE (image; refImage; nomImage)
ECRIRE IMAGE DANS PRESSE PAPIERS (image)
ECRIRE NOM RESSOURCE (typeRes; resID; nomRes{; fichierRes})
Ecrire proprietes groupe (réfGroupe; nom; propriétaire{; membres}) → Numérique
ECRIRE PROPRIETES RESSOURCE (typeRes; resID; attrRes{; fichierRes})
Ecrire proprietes utilisateur (réfUtilisateur; nom; démarrage; motDePasse; nbUtilisation; dernièreUtilisation{; adhésions}) → Numérique
ECRIRE RESSOURCE (typeRes; numRes; donnéesRes{; fichierRes})
ECRIRE RESSOURCE CHAINE (resNum; resDonnées{; resFichier})
ECRIRE RESSOURCE IMAGE (resNum; resDonnées{; resFichier})
ECRIRE RESSOURCE TEXTE (resNum; resDonnées{; resFichier})
ECRIRE TEXTE DANS PRESSE PAPIERS (texte)
ECRIRE VARIABLE PROCESS (process; varDestination; exprSource{; varDestination2; exprSource2; ...; varDestinationN; exprSourceN})
ECRIRE VARIABLES (doc; variable{; variable2; ...; variableN})
ECRITURE ASCII ({table; }document)
ECRITURE DIF ({table; }document)
ECRITURE SYLK ({table; }document)
EFFACER ENSEMBLE (ensemble)
EFFACER FENETRE {(fenêtre)}
EFFACER PRESSE PAPIERS
EFFACER SELECTION (tempo)
EFFACER SEMAPHORE (sémaphore)
EFFACER VARIABLE (variable)
Element parent (liste; réfElément) → Numérique
Element selectionne (liste) → Entier long
EMPILER ENREGISTREMENT {(table)}
En entete → Booléen

En pied → Booléen
En rupture → Booléen
ENDORMIR PROCESS (process; durée)
ENLEVER ELEMENT ({table; }ensemble)
Enregistrement charge ({table}) → Booléen
Enregistrement modifie {(table)} → Booléen
ENREGISTREMENT PRECEDENT {(table)}
ENREGISTREMENT SELECTION {(table)}
ENREGISTREMENT SUIVANT {(table)}
Enregistrement verrouille {(table)} → Booléen
Enregistrements dans ensemble (ensemble) → Numérique
Enregistrements dans table {(table)} → Numérique
Enregistrements trouves {(table)} → Numérique
ENREGISTRER EVENEMENT (message {; importance})
ENREGISTRER IMAGE (document; image)
ENSEMBLE VIDE ({table; }ensemble)
Ent (nombre) → Numérique
ENTIER LONG VERS BLOB (entierLong; blob; ordreOctets{; offset | *})
ENTIER VERS BLOB (entier; blob; ordreOctet{; offset | *})
ENUMERATION VERS TABLEAU (énumération; tableau{; réfEléments})
ENVOYER BLOB HTML (blob; type {; sansContexte})
ENVOYER ENREGISTREMENT {(table)}
ENVOYER FICHIER HTML (fichierHTML)
ENVOYER PAQUET ({docRef; }paquet)
ENVOYER REDIRECTION HTTP (url {; *})
ENVOYER TEXTE HTML (texteHTML{;sansContexte})
ENVOYER VARIABLE (variable)
Est une variable (unPointeur) → Booléen
ETAT ({table; }document{; *})
Etat lecture seulement {(table)} → Booléen
Evenement formulaire → Numérique
Evenement moteur → Entier long
EXECUTER (instruction)
EXECUTER SUR CLIENT (nomClient; nomMéthode {; param1; ...; paramN})
Executer sur serveur(procédure;pile{;nom{;param{; param2; ...; paramN} {; *}}}) → Numérique
Exp (nombre) → Numérique
EXPORTER DONNEES (nomFichier {; projet{ ; * }})

F

Faux → Booléen
Fenetre premier plan {(*)} → RefFen
Fenetre suivante (fenêtre) → Numérique
FERMER DOCUMENT (réfDocument)
FERMER FENETRE {(numFenêtre)}

FERMER FICHIER RESSOURCES (resFichier)
Fichier application → Alpha
Fichier donnees {(segment)} → Alpha
Fichier structure → Alpha
FILTRER EVENEMENT
FILTRER FRAPPE CLAVIER (carFiltré)
Fin de selection {(table)} → Booléen
Fin sous enregistrement (sousTable) → Booléen
FIXER COULEURS RVB ({*; }objet; couleurAvantPlan; couleurArrièrePlan)
FIXER DESTINATION RECHERCHE (destinationType{; destinationObjet})
FIXER ENTETE HTTP (entete | tabChamps {; tabValeurs})
FIXER HISTORIQUE (historique | *)
FIXER INDEX (champ; index{; mode} {; *})
FIXER INTERFACE (interface)
FIXER LIMITE RECHERCHE (limite)
FIXER LIMITES AFFICHAGE WEB (nombreEnr{; nombrePages{; imageRéf}})
FIXER MINUTEUR (nbTicks)
FIXER NIVEAU COMPARAISON REEL (epsilon)
FIXER PAGE ACCUEIL (homePage)
FIXER PARAMETRE BASE ({table; } sélecteur; valeur
FIXER PROFONDEUR ECRAN (profondeur{; couleurs{; écran}})
FIXER RACINE HTML (chemAccèsHTML)
FIXER TAILLE BLOB (blob; taille{; remplisseur})
FIXER TEMPORISATION WEB (timeout)
FIXER TIMEOUT (secondes)
FIXER TITRES CHAMPS (table | sous-table; titresChamps; numChamps)
FIXER TITRES TABLES (titresTables; numTables)
FORMULAIRE ENTREE ({table; }formulaire{; *})
FORMULAIRE SORTIE ({table; }formulaire)
Frappe clavier → Alpha

G

GENERER CLES CRYPTAGE (cléPrivée; cléPublique{; longueur})
GENERER CLIC (sourisX; sourisY{; process}{; *})
GENERER DEMANDE CERTIFICAT(cléPrivée; demCertif; tabCodes; tabNoms)
GENERER EVENEMENT (quoi; message; quand; sourisX; sourisY; modifiers{; process})
GENERER FRAPPE CLAVIER (code{; modifiers{; process}})
Gestalt (sélecteur; valeur) → Numérique
GRAPHE (graphZone; graphNum; xCatégories; zValeurs{; zValeurs2; ...; zValeursN})
GRAPHE SUR SELECTION ({table; }numGraphe; axeX; axeZ{; axeZ2; ...; axeZN})

H

Hasard → Numérique
Hauteur barre de menus → Entier long
Hauteur ecran {(*)} → Numérique

Heure (chaineHeure) → Heure
Heure courante {(*)} → Heure

I

IMAGE VERS BLOB (image; blobImage; format)
IMAGE VERS GIF (imagePICT; blobGIF)
IMPORTER DONNEES (nomFichier {; projet{ ; * }})
IMPRESSION ECRAN (écran)
IMPRIMER ENREGISTREMENT ({table}{; }{* | >})
IMPRIMER ETIQUETTES ({table}{; étiqFichier}{; * | >})
IMPRIMER LIGNE ({table; }form)
IMPRIMER SELECTION ({table}{; }{* | >})
INACTIVER BOUTON ({*; }objet)
INACTIVER LIGNE MENU (menu; ligneMenu{; process})
Indefinie (variable) → Booléen
INFORMATION ELEMENT (liste; positionElém; numElém; textElém{; sous-liste{; déployé}})
INFORMATIONS PROCESS (process; procNom; procStatut; procTemps{; procVisible{; uniqueID{; origine}}})
INSCRIRE CLIENT (nomClient {; période} {; *})
Inserer chaine (source; insertion; position) → Alpha
INSERER DANS BLOB (blob; offset; nombre{; défaut})
INSERER ELEMENT (liste; avantElément | *; texteElément; numElément{; sous-liste; déployé})
INSERER LIGNE MENU (menu; aprèsLigne; texteLigne{; process})
INSERER LIGNES (tableau; position{; combien})
INTERSECTION (ensemble1; ensemble2; résultat)
INVERSER FOND ({*; }textVar | textChp)
ISO vers Mac (texte) → Chaîne

J

JOINTURE (tableN; table1)
JOUER SON (nomObjet{; canal})
Jour de (date) → Numérique

L

LAISSER MESSAGES
LANCER SERVEUR WEB
Largeur ecran {(*)} → Numérique
LECTURE ASCII ({table; }document)
LECTURE DIF ({table; }document)
LECTURE ECRITURE {(table | *)}
LECTURE SEULEMENT {(table | *)}
LECTURE SYLK ({table; }document)
LIBERER ENREGISTREMENT {(table)}
LIEN RETOUR (table1 | champ1)

LIENS AUTOMATIQUES (lienAppel{; lienRetour})
Lire chaine dans liste (resNum; strNum{; resFichier}) → Alpha
LIRE CLIENTS INSCRITS (listeClients; nbMéthodes)
LIRE ENTETE HTTP (entete | tabChamps{; tabValeurs})
LIRE FICHIER IMAGE (nomFichier; image)
LIRE ICONE DOCUMENT(cheminAccès; icône{; taille})
Lire ID ressource composant (nomComp; typeRes; numResOriginal) → Numérique
LIRE IMAGE DANS BIBLIOTHEQUE (refImage; image)
LIRE IMAGE DANS PRESSE PAPIERS (image)
LIRE INFORMATIONS SERIALISATION(clé; utilisateur; société; connectés; maxUtilisateurs)
Lire interface → Numérique
LIRE LISTE GROUPE (nomsGroupe; numérosGroupe)
LIRE LISTE UTILISATEURS (nomsUtil; numérosUtil)
Lire nom ressource (typeRes; resID{; fichierRes}) → Alpha
Lire parametre base ({table; } sélecteur) → Entier long
LIRE PRESSE PAPIERS (typeDonnées; données)
LIRE PROPRIETES BLOB (blob; compressé{; tailleDécompressée{; tailleCourante}})
LIRE PROPRIETES CHAMP (chpPtr | tableNum{; champNum}; champType{; champLong
{; indexé{; unique{; invisible}}}})
LIRE PROPRIETES ELEMENT (liste; réfElément; saisissable{; styles{; icône}})
LIRE PROPRIETES FORMULAIRE ({table;} nomForm; largeur; hauteur{; nbPages{; largeurFixe
{; hauteurFixe{; titre}}}})
LIRE PROPRIETES GROUPE (réfGroupe; nom; propriétaire{; membres})
LIRE PROPRIETES LIEN (chpPtr | tableNum{; champNum}; tableDest; champDest
{; discriminant{; allerAuto{; retourAuto}}})
LIRE PROPRIETES LISTE (liste; apparence{; icône{; ligneHauteur{; doubleClic}}})
Lire proprietes ressource (typeRes; resID{; fichierRes}) → Numérique
LIRE PROPRIETES SAISIE CHAMP (chpPtr | tableNum{; champNum}; nomEnum; obligatoire;
nonSaisissable; nonModifiable)
LIRE PROPRIETES TABLE (tablePtr | tableNum; invisible{; trigSvgdeNouv{; trigSvgdeEnr
{; trigSupprEnr{; trigChargEnr}}}})
LIRE PROPRIETES UTILISATEUR (réfUtilisateur; nom; démarrage; motDePasse; nbUtilisations;
dernièreUtilisation{; adhésions})
LIRE RECT OBJET ({ *; } objet; gauche; haut; droite; bas)
LIRE RESSOURCE (typeRes; numRes; donnéesRes{; fichierRes})
Lire ressource chaine (resNum{; resFichier}) → Alpha
LIRE RESSOURCE ICONE (resNum; imageDest{; resFichier})
LIRE RESSOURCE IMAGE (resNum; resDonnées{; resFichier})
Lire ressource texte (resNum{; resFichier}) → Texte
Lire texte dans presse papiers → Alpha
Lire texte edite → Texte
LIRE VARIABLE PROCESS (process; varSource; varDestination{; varSource2; varDestination2; ...;
varSourceN; varDestinationN})
LIRE VARIABLES (doc; variable{; variable2; ...; variableN})
LIRE VARIABLES FORMULAIRE WEB (tabNoms;tabValeurs)
LISTE DE CHAINES VERS TABLEAU (resNum; chaînes{; resFichier})
LISTE DES DOCUMENTS (cheminAccès; documents)

LISTE DES DOSSIERS (cheminAccès; dossiers)
LISTE DES POLICES (polices)
LISTE DES VOLUMES (volumes)
Liste existante (liste) → Booléen
LISTE FENETRES (fenêtres{; *})
LISTE IMAGES DANS BIBLIOTHEQUE (RefsImages; NomsImages)
LISTE RESSOURCES (resType; resNums; resNoms{; resFichier})
LISTE SEGMENTS DE DONNEES (segments)
LISTE TYPES IMAGES (tabFormats{; tabNoms})
LISTE TYPES RESSOURCE (resTypes{; resFichier})
LISTE VERS BLOB (liste; blob{; offset | *})
Log (nombre) → Numérique
Longueur (chaîne) → Numérique

M

Mac vers ISO (texte) → Chaîne
Mac vers Windows (texte) → Chaîne
Macintosh commande enfoncee → Booléen
Macintosh control enfoncee → Booléen
Macintosh option enfoncee → Booléen
Majusc (chaîne) → Alpha
Majuscule enfoncee → Booléen
Marque ligne menu (menu; ligneMenu{; process}) → Alpha
MARQUER ENREGISTREMENTS ({nomEnsemble})
MARQUER LIGNE MENU (menu; ligneMenu; marque{; process})
Max (séries) → Numérique
MAXIMISER FENETRE {(fenêtre)}
Menu choisi → Numérique
MESSAGE (message)
Min (séries) → Numérique
MINIMISER FENETRE {(fenêtre)}
Minusc (chaîne) → Alpha
Modifie (champ) → Booléen
MODIFIER ENREGISTREMENT ({table}{; }{*})
MODIFIER SELECTION ({table}{; *}{; *})
MODIFIER SOUS ENREGISTREMENT (sousTable; formulaire{; *})
Modulo (nombre1; nombre2) → Numérique
Mois de (date) → Numérique
MONTRER PROCESS (process)
Moyenne (séries) → Numérique

N

NE PAS VALIDER
Nil (unPointeur) → Booléen
Niveau → Numérique
Niveau du trigger → Numérique
NIVEAUX DE RUPTURES (niveau{; sautPage})
Nom commande (commande) → Alpha
Nom de la machine → Alpha
Nom de la table (numTable | ptrTable) → Alpha
Nom de police (numéro) → Alpha
Nom du champ (champPtr) | (tableNum; champNum) → Alpha
Nom du possesseur → Alpha
Nom methode courante → Alpha
Nombre de champs (tableNum | tablePtr) → Numérique
Nombre de lignes du menu (menu{; process}) → Numérique
Nombre de menus {(process)} → Numérique
Nombre de millisecondes → Entier long
Nombre de parametres → Numérique
Nombre de process → Entier
Nombre de process utilisateurs → Entier
Nombre de tables → Numérique
Nombre de ticks → Numérique
Nombre ecrans → Entier long
Nombre elements (liste) → Entier long
Nombre utilisateurs → Entier
NOMMER ENSEMBLE ({table; }ensemble)
Non (booléen) → Booléen
Nouveau process (méthode; pile{; nom{; param{; param2; ...; paramN}{; *}}}) → Numérique
Nouvel enregistrement ({table}) → Booléen
Nouvelle liste → RéfListe
Num (expression) → Numérique
Numero dans selection {(table)} → Numérique
Numero de police (nomPolice) → Entier long
Numero du jour (date) → Numérique
Numero du process courant → Numérique
Numero enregistrement {(table)} → Numérique
Numerotation automatique {(table)} → Numérique

O

Ouvrir document (nomDoc{; type{; mode}}) → docRef
Ouvrir fichier ressources (resNomFichier{; typeFichier}) → docRef
OUVRIR URL WEB (url {; *})

P

Page formulaire courante → Numérique
Page impression → Numérique
PAGE PRECEDENTE
PAGE SUIVANTE
PARAMETRES DU GRAPHE (graphe; xmin; xmax; ymin; ymax; xprop; grilleX; grilleY; titre {; titre2; ...; titreN})
PARAMETRES IMPRESSION
PAS DE TRACE
PASSER AU PREMIER PLAN (process)
Pendant → Booléen
Pointeur vers (nomVar) → Pointeur
Pop up menu (contenu{; parDéfaut}) → Numérique
Position (àChercher; chaîne) → Numérique
Position dans document (docRef) → Numérique
Position deposer → Numérique
Position element liste (liste; réfElément) → Numérique
POSITION MESSAGE (x; y{; fenêtre})
POSITION SOURIS (sourisX; sourisY; boutonSouris{; *})
PREMIERE PAGE
Process de la fenetre {(fenêtre)} → Numérique
Process de premier plan {(*)} → Entier
Process interrompu → Booléen
PROFONDEUR ECRAN (profondeur; couleurs{; écran})
PROPRIETES DOCUMENT (document; verrouillé; invisible; créé le; créé à ; modifié le; modifié à)
PROPRIETES DU TRIGGER (niveauTrigger; evenementBase; tableNum; enregNum)
PROPRIETES DU VOLUME (volume; taille; utilisé; libre)
PROPRIETES GLISSER DEPOSER (srcObjet; srcElément; srcProcess)
PROPRIETES IMAGE (image; largeur; hauteur{; hOffset{; vOffset{; mode}}})
PROPRIETES PLATE FORME (plate-forme{; système{; machine}})

Q

QUITTER 4D

R

Raccourci clavier (menu; ligneMenu{; process}) → Numérique
Racine carree (nombre) → Numérique
REACTIVER PROCESS (process)
RECEVOIR BUFFER (varRéception)
RECEVOIR ENREGISTREMENT {(table)}
RECEVOIR PAQUET ({docRef; }réceptVar; stopCar | nbCar)
RECEVOIR VARIABLE (variable)
REDESSINER (objet)
REDESSINER FENETRE {(fenêtre)}
REDESSINER LISTE (liste)
REDUIRE SELECTION ({table; }nombre)
REEL VERS BLOB (réel; blob; formatRéel{; offset | *})
REFUSER {(champ)}
REGLER SERIE (port | opération{; param | document})
Remplacer caracteres (source; nouveau; position) → Alpha
Remplacer chaine (source; obsolète; nouveau{; remplacements}) → Alpha
RESOUDRE ALIAS (cheminAlias; cheminCible)
RESOUDRE POINTEUR (pointeur; nomVar; numTable; numChamp)
REUNION (ensemble1; ensemble2; résultat)

S

SAUT DE PAGE {(* | >)}
SCAN INDEX (champ; nombre{; > ou <})
SELECTION LIMITEE VERS TABLEAU (début; fin; champ | table; tableau{; champ2 | table2; tableau2; ...; champN | tableN; tableauN})
SELECTION RETOUR (champ)
SELECTION VERS TABLEAU (champ | table; tableau{; champ2 | table2; tableau2; ...; champN | tableN; tableauN})
Selectionner dossier ({message}) → Alpha
SELECTIONNER ELEMENT (liste; positionElém)
SELECTIONNER ELEMENT PAR REFERENCE (liste; réfElément)
SELECTIONNER TEXTE (zone; débutSél; finSél)
Self → Pointeur
Semaphore (sémaphore {; nbTicks}) → Booléen
SIECLE PAR DEFAUT (siècle{; anPivot})
Sin (nombre) → Numérique
Somme (séries) → Numérique
Somme des carres (séries) → Numérique
Sous chaine (source; àPartirDe{; nbCars}) → Alpha
SOUS ENREGISTREMENT PRECEDENT (sousTable)
SOUS ENREGISTREMENT SUIVANT (sousTable)
Sous enregistrements trouves (subtable) → Numérique
Sous total (valeurs{; sautPage}) → Numérique

STATISTIQUES DU CACHE WEB (pages; hits; usage)
Statut du process (process) → Numérique
STOCKER ANCIEN (champ)
STOCKER ENREGISTREMENT {(table)}
STOCKER ENSEMBLE (ensemble; document)
STOCKER LISTE (liste; nomListe)
STOCKER SUR LIEN (champ)
STOP
Style ligne menu (menu; ligneMenu{; process}) → Numérique
Supprimer chaine (source; position; nombreCar) → Alpha
SUPPRIMER DANS BLOB (blob; offset; nombre)
SUPPRIMER DOCUMENT (document)
SUPPRIMER DOSSIER (dossier)
SUPPRIMER ELEMENT (liste; élémentRef | *{; *})
SUPPRIMER ENREGISTREMENT {(table)}
SUPPRIMER IMAGE DANS BIBLIOTHEQUE (refImage)
SUPPRIMER LIGNE MENU (menu; ligneMenu{; process})
SUPPRIMER LIGNES (tableau; position{; combien})
SUPPRIMER LISTE (liste{; *})
SUPPRIMER MESSAGES
SUPPRIMER RESSOURCE (resType; resNum{; resFichier})
SUPPRIMER SELECTION {(table)}
SUPPRIMER SOUS ENREGISTREMENT (sousTable)
SUPPRIMER UTILISATEUR (numUtilisateur)
SUSPENDRE PROCESS (process)

T

Table (numTable | unPtr) → Num | Pointeur
Table du formulaire courant → Pointeur
TABLE PAR DEFAUT (table)
Table par defaut courante → Pointeur
TABLEAU ALPHA (longueurChaîne; nomTableau; taille{; taille2})
TABLEAU BOOLEEN (nomTableau; taille{; taille2})
TABLEAU BOOLEEN SUR ENSEMBLE (tabBooléen {; ensemble})
TABLEAU DATE (nomTableau; taille{; taille2})
TABLEAU ENTIER (nomTableau; taille{; taille2})
TABLEAU ENTIER LONG (nomTableau; taille{; taille2})
TABLEAU ENTIER LONG SUR SELECTION (Table; tabEnrg {; nomSélection})
TABLEAU IMAGE (nomTableau; taille{; taille2})
TABLEAU POINTEUR (nomTableau; taille{; taille2})
TABLEAU REEL (nomTableau; taille{; taille2})
TABLEAU TEXTE (nomTableau; taille{; taille2})
TABLEAU VERS ENUMERATION (tableau; énumération{; réfEléments})
TABLEAU VERS LISTE DE CHAINES (chaînes; resNum{; resFichier})
TABLEAU VERS SELECTION (tableau; champ{; tableau2; champ2; ...; tableauN; champN})
Taille BLOB (blob) → Entier long

Taille document (document{; *}) → Numérique
Taille image (image) → Numérique
Taille tableau (tableau) → Numérique
Tan (nombre) → Numérique
Tester chemin acces (cheminAccès) → Numérique
Tester presse papiers (typeDonnées) → Numérique
Tester semaphore (semaphore) → Booléen
Texte ligne menu (menu; ligneMenu{; process}) → Alpha
TEXTE SELECTIONNE (zone; débutSél; finSél)
TEXTE VERS BLOB (texte; blob; formatTexte{; offset | *})
TITRE BOUTON ({*; }objet; libellBouton)
Titre fenetre {(fenêtre)} → Chaîne
Titre menu (menu{; process}) → Alpha
TOUS LES SOUS ENREGISTREMENTS (sousTable)
TOUT SELECTIONNER {(table)}
TRACE
Transaction en cours → Booléen
TRIER ({table}{; champ}{; > ou <}{; champ2; > ou <2; ...; champN; > ou <N}{; *})
TRIER LISTE (liste{; > ou <})
TRIER PAR FORMULE ({table}{; expression}{; > ou <}{; expression2; > ou <2; ...; expressionN; > ou <N})
TRIER SOUS ENREGISTREMENTS (sousTable; sousChamp{; direction}{; sousChamp2; direction2; ...; sousChampN; directionN})
TRIER SUR INDEX
TRIER TABLEAU (tableau{; tableau2; ...; tableauN}{; sensDuTri})
Troncature (nombre; nbDécimales) → Numérique
Trouver clef index (champIndexé; valeur) → Entier long
Type (champVar) → Numérique
Type application → Entier long
Type document (document) → Alpha
Type fenetre {(fenêtre)}
Type version → Entier long

U

Utilisateur courant → Alpha
Utilisateur supprime (numUtilisateur) → Booléen
UTILISER ENSEMBLE (ensemble)
UTILISER FILTRE (filtre | *{; typeFiltre})
UTILISER PARAMETRES IMPRESSION ({table; }formulaire)
UTILISER SELECTION (tempo)

V

VALEURS DISTINCTES (champ; tableau)
VALIDER
Valider mot de passe (numUtilisateur; motDePasse) → Booléen
VALIDER TRANSACTION
VARIABLE VERS BLOB (variable; blob{; *})
VARIABLE VERS VARIABLE (process; varDestination; varSource{; varDestination2; varSource2; ...; varDestinationN; varSourceN})
Variance (séries) → Numérique
Verrouillage majuscule enfoncee → Booléen
VERROUILLE PAR ({table; }process; utilisateur; machine; nomProcess)
Version application {(*)} → Alpha
VISUALISER SELECTION ({table}{; *}; *)
Vrai → Booléen

W

Windows Alt enfoncee → Booléen
Windows Ctrl enfoncee → Booléen
Windows vers Mac (texte) → Chaîne

Constantes

BLOB

Constante	Type	Valeur
Chaîne en C	Entier long	0
Chaîne pascal	Entier long	1
Format réel double Macintosh	Entier long	2
Format réel double PC	Entier long	3
Format réel étendu	Entier long	1
Format réel natif	Entier long	0
Méthode de compression compacte	Entier long	1
Méthode de compression rapide	Entier long	2
Non compressé	Entier long	0
Ordre octets Macintosh	Entier long	1
Ordre octets natif	Entier long	0
Ordre octets PC	Entier long	2
Texte avec longueur	Entier long	2
Texte sans longueur	Entier long	3

Caractères latins ISO

Constante	Type	Valeur
ISO L1 0 majus circonflexe	Chaîne	Ô
ISO L1 a aigu	Chaîne	á
ISO L1 a circonflexe	Chaîne	â
ISO L1 a grave	Chaîne	à
ISO L1 A majus aigu	Chaîne	Á
ISO L1 A majus circonflexe	Chaîne	Â
ISO L1 A majus grave	Chaîne	À
ISO L1 A majus tilde	Chaîne	Ã
ISO L1 A majus umlaut	Chaîne	Å
ISO L1 A majus umlaut	Chaîne	Ä
ISO L1 a rond	Chaîne	å
ISO L1 a tilde	Chaîne	ã
ISO L1 a umlaut	Chaîne	ä
ISO L1 ae ligature	Chaîne	æ
ISO L1 AE majus ligature	Chaîne	&AELig;
ISO L1 c cédille	Chaîne	ç
ISO L1 C majus cédille	Chaîne	Ç
ISO L1 Copyright	Chaîne	©
ISO L1 e aigu	Chaîne	é
ISO L1 e circonflexe	Chaîne	ê
ISO L1 e grave	Chaîne	è
ISO L1 E majus aigu	Chaîne	É
ISO L1 E majus circonflexe	Chaîne	Ê
ISO L1 E majus grave	Chaîne	È
ISO L1 E majus unlaut	Chaîne	Ë
ISO L1 e umlaut	Chaîne	ë
ISO L1 Es zett allemand	Chaîne	ß
ISO L1 Et commercial	Chaîne	&
ISO L1 eth islandais	Chaîne	ð
ISO L1 Eth majus islandais	Chaîne	Ð
ISO L1 Guillemets	Chaîne	"
ISO L1 i aigu	Chaîne	í
ISO L1 i circonflexe	Chaîne	î
ISO L1 i grave	Chaîne	ì
ISO L1 I majus aigu	Chaîne	Í

Caractères latins ISO (suite)

Constante	Type	Valeur
ISO L1 I majus circonflex	Chaîne	Î
ISO L1 I majus grave	Chaîne	Ì
ISO L1 I majus umlaut	Chaîne	Ï
ISO L1 i umlaut	Chaîne	ï
ISO L1 Inférieur	Chaîne	<
ISO L1 Marque déposée	Chaîne	®
ISO L1 N majus tilde	Chaîne	Ñ
ISO L1 n tilde	Chaîne	ñ
ISO L1 o aigu	Chaîne	ó
ISO L1 o barré	Chaîne	ø
ISO L1 o circonflexe	Chaîne	ô
ISO L1 o grave	Chaîne	ò
ISO L1 O majus aigu	Chaîne	Ó
ISO L1 O majus barré	Chaîne	Ø
ISO L1 O majus grave	Chaîne	Ò
ISO L1 O majus tilde	Chaîne	Õ
ISO L1 O majus umlaut	Chaîne	Ö
ISO L1 o tilde	Chaîne	õ
ISO L1 o umlaut	Chaîne	ö
ISO L1 Supérieur	Chaîne	>
ISO L1 thorn islandais	Chaîne	þ
ISO L1 THORN majus islandais	Chaîne	Þ
ISO L1 u aigu	Chaîne	ú
ISO L1 u circonflexe	Chaîne	û
ISO L1 u grave	Chaîne	ù
ISO L1 U majus aigu	Chaîne	Ú
ISO L1 U majus circonflexe	Chaîne	Û
ISO L1 U majus grave	Chaîne	Ù
ISO L1 U majus umlaut	Chaîne	Ü
ISO L1 u umlaut	Chaîne	ü
ISO L1 y aigu	Chaîne	ý
ISO L1 Y majus aigu	Chaîne	Ý
ISO L1 y umlaut	Chaîne	ÿ

Chercher fenetre

Constante	Type	Valeur
Dans barre de menu	Entier long	1
Dans barre de titre	Entier long	4
Dans case de contrôle de taille	Entier long	5
Dans case de fermeture	Entier long	6
Dans case de zoom	Entier long	8
Dans fenêtre système	Entier long	2
Dans zone contenu	Entier long	3

Codes ASCII

Constante	Type	Valeur
Apostrophe	Entier long	39
Arobase	Entier long	64
ASCII ACK	Entier long	6
ASCII BEL	Entier long	7
ASCII BS	Entier long	8
ASCII CAN	Entier long	24
ASCII CR	Entier long	13
ASCII DC1	Entier long	17
ASCII DC2	Entier long	18
ASCII DC3	Entier long	19
ASCII DC4	Entier long	20
ASCII DEL	Entier long	127
ASCII DLE	Entier long	16
ASCII EM	Entier long	25
ASCII ENQ	Entier long	5
ASCII EOT	Entier long	4
ASCII ESC	Entier long	27
ASCII ETB	Entier long	23
ASCII ETX	Entier long	3
ASCII FF	Entier long	12
ASCII FS	Entier long	28
ASCII GS	Entier long	29
ASCII HT	Entier long	9
ASCII LF	Entier long	10
ASCII NAK	Entier long	21
ASCII NUL	Entier long	0
ASCII RS	Entier long	30
ASCII SI	Entier long	15
ASCII SO	Entier long	14
ASCII SOH	Entier long	1
ASCII SP	Entier long	32
ASCII STX	Entier long	2
ASCII SUB	Entier long	26
ASCII SYN	Entier long	22
ASCII US	Entier long	31

Codes ASCII (suite)

Constante	Type	Valeur
ASCII VT	Entier long	11
Echappement	Entier long	27
Effacement arrière	Entier long	8
Entrée	Entier long	3
Espace insécable	Entier long	202
Espacement	Entier long	32
Guillemets	Entier long	34
Point	Entier long	46
Retour à la ligne	Entier long	10
Retour chariot	Entier long	13
Tabulation	Entier long	9

Communications

Constante	Type	Valeur
Bit de stop deux	Entier long	-16384
Bit de stop un	Entier long	16384
Bit de stop un et demi	Entier long	-32768
Bits de données 5	Entier long	0
Bits de données 6	Entier long	2048
Bits de données 7	Entier long	1024
Bits de données 8	Entier long	3072
Parité impaire	Entier long	4096
Parité paire	Entier long	12288
Pas de parité	Entier long	0
Port imprimante MacOS	Entier long	0
Port série MacOS	Entier long	1
Protocole aucun	Entier long	0
Protocole DTR	Entier long	30
Protocole XONXOFF	Entier long	20
Vitesse 115200	Entier long	1022
Vitesse 1200	Entier long	94
Vitesse 1800	Entier long	62
Vitesse 19200	Entier long	4
Vitesse 230400	Entier long	1021
Vitesse 2400	Entier long	46
Vitesse 300	Entier long	380
Vitesse 3600	Entier long	30
Vitesse 4800	Entier long	22
Vitesse 57600	Entier long	0
Vitesse 600	Entier long	189
Vitesse 7200	Entier long	14
Vitesse 9600	Entier long	10

Couleurs

Constante	Type	Valeur
Blanc	Entier long	0
Bleu	Entier long	6
Bleu clair	Entier long	7
Bleu foncé	Entier long	5
Gris	Entier long	14
Gris clair	Entier long	12
Gris foncé	Entier long	11
Jaune	Entier long	1
Marron	Entier long	13
Marron foncé	Entier long	10
Noir	Entier long	15
Orange	Entier long	2
Rouge	Entier long	3
Vert	Entier long	8
Vert foncé	Entier long	9
Violet	Entier long	4

Creer fenetre

Constante	Type	Valeur
Avec barre de titre active	Entier long	1
Avec case de contrôle de taille	Entier long	4
Avec case de zoom	Entier long	8
Avec titre de fenêtre	Entier long	2
Dialogue modal	Entier long	1
Dialogue modal déplaçable	Entier long	5
Dialogue ombré	Entier long	3
Dialogue simple	Entier long	2
Fenêtre à coins arrondis	Entier long	16
Fenêtre palette	Entier long	1984
Fenêtre standard	Entier long	8
Fenêtre standard de taille fixe	Entier long	4
Fenêtre standard sans zoom	Entier long	0

Destinations de recherche

Constante	Type	Valeur
Vers ensemble	Entier long	1
Vers sélection courante	Entier long	0
Vers sélection temporaire	Entier long	2
Vers variable	Entier long	3

Documents système

Constante	Type	Valeur
Est un document	Entier long	1
Est un répertoire	Entier long	0
Lecture et écriture	Entier long	0
Lire chemin accès	Entier long	3
Mode écriture	Entier long	1
Mode lecture	Entier long	2

Dossier Système

Constante	Type	Valeur
Extensions Mac	Entier long	10
Menu pomme ou Démarrer	Entier long	9
Menu pomme ou Démarrer_Tous	Entier long	8
Ouverture au démarrage	Entier long	5
Ouverture au démarrage_Tous	Entier long	4
Ouverture extinction Mac	Entier long	7
Ouverture extinction Mac_Tous	Entier long	6
Polices	Entier long	1
Préférences ou Profiles	Entier long	3
Préférences ou Profiles_Tous	Entier long	2
Système	Entier long	0
Tableaux de bord Mac	Entier long	11

Environnement 4D

Constante	Type	Valeur
4D Client	Entier long	4
4D Engine	Entier long	1
4D First	Entier long	6
4D Runtime	Entier long	2
4D Runtime Classic	Entier long	3
4D Server	Entier long	5
4e Dimension	Entier long	0
Version de démonstration	Entier long	1
Version standard	Entier long	0

Euro monnaies

Constante	Type	Valeur
Escudo portugais	Chaîne	PTE
Euro	Chaîne	EUR
Florin néerlandais	Chaîne	NLG
Franc belge	Chaîne	BEF
Franc français	Chaîne	FRF
Franc luxembourgeois	Chaîne	LUF
Lire italienne	Chaîne	ITL
Livre irlandaise	Chaîne	IEP
Mark allemand	Chaîne	DEM
Mark finlandais	Chaîne	FIM
Peseta espagnole	Chaîne	ESP
Schilling autrichien	Chaîne	ATS

Evénements (Modifiers)

Constante	Type	Valeur
Bit activation fenêtre	Entier long	0
Bit bouton souris	Entier long	7
Bit touche commande	Entier long	8
Bit touche contrôle	Entier long	12
Bit touche contrôle droite	Entier long	15
Bit touche majuscule	Entier long	9
Bit touche majuscule droite	Entier long	13
Bit touche option	Entier long	11
Bit touche option droite	Entier long	14
Bit touche verrouillage maj	Entier long	10
Masque activation fenêtre	Entier long	1
Masque bouton souris	Entier long	128
Masque touche commande	Entier long	256
Masque touche contrôle	Entier long	4096
Masque touche contrôle droite	Entier long	32768
Masque touche majuscule	Entier long	512
Masque touche majuscule droite	Entier long	8192
Masque touche option	Entier long	2048
Masque touche option droite	Entier long	16384
Masque touche verrouillage maj	Entier long	1024

Evénements (types)

Constante	Type	Valeur
Activation fenêtre	Entier long	8
Bouton souris enfoncé	Entier long	1
Bouton souris relâché	Entier long	2
Evénement disque	Entier long	7
Evénement nul	Entier long	0
Evénement système	Entier long	15
Mise à jour fenêtre	Entier long	6
Répétition touche	Entier long	5
Touche enfoncée	Entier long	3
Touche relâchée	Entier long	4

Evénements formulaire

Constante	Type	Valeur
Sur activation	Entier long	11
Sur affichage corps	Entier long	8
Sur appel extérieur	Entier long	10
Sur appel zone du plug in	Entier long	19
Sur après frappe clavier	Entier long	28
Sur avant frappe clavier	Entier long	17
Sur case de fermeture	Entier long	22
Sur chargement	Entier long	1
Sur clic souris	Entier long	4
Sur déposer	Entier long	16
Sur désactivation	Entier long	12
Sur données modifiées	Entier long	20
Sur double clic souris	Entier long	13
Sur entête	Entier long	5
Sur fermeture corps	Entier long	26
Sur gain focus	Entier long	15
Sur glisser	Entier long	21
Sur impression corps	Entier long	23
Sur impression pied de page	Entier long	7
Sur impressions sous total	Entier long	6
Sur libération	Entier long	24
Sur menu sélectionné	Entier long	18
Sur minuteur	Entier long	27
Sur ouverture corps	Entier long	25
Sur perte focus	Entier long	14
Sur redimensionnement	Entier long	29
Sur validation	Entier long	3

Evénements moteur

Constante	Type	Valeur
Sur chargement enregistrement	Entier long	4
Sur sauvegarde enregistrement	Entier long	2
Sur sauvegarde nouvel enreg	Entier long	1
Sur suppression enregistrement	Entier long	3

Expressions

Constante	Type	Valeur
MAXENT	Entier long	32767
MAXLARGTEXTE	Entier long	32000
MAXLONG	Entier long	2147483647

FIXER COULEUR RVB

Constante	Type	Valeur
Couleur claire par défaut	Entier long	-4
Couleur par défaut arrière plan	Entier long	-2
Couleur par défaut premier plan	Entier long	-1
Couleur sombre par défaut	Entier long	-3

Fonctions mathématiques

Constante	Type	Valeur
Degré	Numérique	0.0174532925199432958
Nombre e	Numérique	2.71828182845904524
Pi	Numérique	3.141592653589793239
Radian	Numérique	57.29577951308232088

Formats d'affichage des dates

Constante	Type	Valeur
Abrégé Jour Mois Année	Entier long	6
Format abrégé	Entier long	2
Format court	Entier long	1
Format long	Entier long	3
Format spécial	Entier long	4
Jour Mois Année	Entier long	5
Spécial forcé	Entier long	7

Formats d'affichage des heures

Constante	Type	Valeur
h mn	Entier long	2
h mn Matin Après Midi	Entier long	5
h mn s	Entier long	1
Heure Minute	Entier long	4
Heure Minute Seconde	Entier long	3

Formats d'affichage des images

Constante	Type	Valeur
Non tronquée	Entier long	2
Proportionnelle	Entier long	5
Proportionnelle centrée	Entier long	6
Sur fond	Entier long	3
Tronquée centrée	Entier long	1
Tronquée non centrée	Entier long	4

Interface de plate-forme

Constante	Type	Valeur
Mac OS 7	Entier long	0
Plate-forme automatique	Entier long	-1
Platinum	Entier long	3
Thème Mac	Entier long	4
Windows 9x	Entier long	2
Windows NT 3.51	Entier long	1

Journal d'événements Windows NT

Constante	Type	Valeur
Message d’avertissement	Entier long	1
Message d’erreur	Entier long	2
Message d’information	Entier long	0

Jours et mois

Constante	Type	Valeur
Août	Entier long	8
Avril	Entier long	4
Décembre	Entier long	12
Dimanche	Entier long	1
Février	Entier long	2
Janvier	Entier long	1
Jeudi	Entier long	5
Juillet	Entier long	7
Juin	Entier long	6
Lundi	Entier long	2
Mai	Entier long	5
Mardi	Entier long	3
Mars	Entier long	3
Mercredi	Entier long	4
Novembre	Entier long	11
Octobre	Entier long	10
Samedi	Entier long	7
Septembre	Entier long	9
Vendredi	Entier long	6

Listes hiérarchiques

Constante	Type	Valeur
A la Macintosh	Entier long	1
A la Windows	Entier long	2
Réf icône Macintosh	Entier long	860
Réf icône Windows	Entier long	138
Utiliser réf image	Entier long	131072
Utiliser ressource PICT	Entier long	65536

Moteur de la base

Constante	Type	Valeur
Aucun enregistrement courant	Entier long	-1
Est un nouvel enregistrement	Entier long	-3

Numéros de port TCP

Constante	Type	Valeur
TCP Authentication	Entier long	113
TCP DNS	Entier long	53
TCP Finger	Entier long	79
TCP FTP Control	Entier long	21
TCP FTP Data	Entier long	20
TCP Gopher	Entier long	70
TCP HTTP WWW	Entier long	80
TCP IMAP3	Entier long	220
TCP Kerberos	Entier long	88
TCP KLogin	Entier long	543
TCP Nickname	Entier long	43
TCP NNTP	Entier long	119
TCP NTalk	Entier long	518
TCP NTP	Entier long	123
TCP PMCP	Entier long	1643
TCP PMD	Entier long	1642
TCP POP3	Entier long	110
TCP Printer	Entier long	515
TCP RADACCT	Entier long	1646
TCP RADIUS	Entier long	1645
TCP Remote Cmd	Entier long	514
TCP Remote Exec	Entier long	512
TCP Remote Login	Entier long	513
TCP Router	Entier long	520
TCP SMTP	Entier long	25
TCP SNMP	Entier long	161
TCP SNMPTRAP	Entier long	162
TCP SUN RPC	Entier long	111
TCP Talk	Entier long	517
TCP Telnet	Entier long	23
TCP TFTP	Entier long	69
TCP UUCP	Entier long	540
TCP UUCP RLOGIN	Entier long	541

Paramètres de la base

Constante	Type	Valeur
Adresse IP d'écoute	Entier long	16
Appels système 4D Client	Entier long	12
Appels système 4D Server	Entier long	11
Appels système 4e Dimension	Entier long	10
Compression index	Entier long	4
Jeu de caractères	Entier long	17
Maximum process Web	Entier long	7
Minimum process Web	Entier long	6
Mode conversion Web	Entier long	8
Numéro du port	Entier long	15
Optimisation accès séq	Entier long	2
Process Web simultanés maxi	Entier long	18
Ratio chercher dans sélec séq	Entier long	5
Ratio de tri séq	Entier long	1
Ratio valeurs distinctes séq	Entier long	3
Taille cache données	Entier long	9
Timeout 4D Client	Entier long	14
Timeout 4D Server	Entier long	13

Position fenêtre

Constante	Type	Valeur
A droite	Entier long	196608
A gauche	Entier long	131072
Centrée horizontalement	Entier long	65536
Centrée verticalement	Entier long	262144
En bas	Entier long	393216
En haut	Entier long	327680

Presse-Papiers

Constante	Type	Valeur
Données absentes presse papiers	Entier long	-102
Données image	Chaîne	PICT
Données texte	Chaîne	TEXT

PROFONDEUR ECRAN

Constante	Type	Valeur
Deux cent cinquante six coul	Entier long	8
Est en couleurs	Entier long	1
Est en niveaux de gris	Entier long	0
Milliers de couleurs	Entier long	16
Millions de couleurs 24 bits	Entier long	24
Millions de couleurs 32 bits	Entier long	32
Noir et blanc	Entier long	0
Quatre couleurs	Entier long	2
Seize couleurs	Entier long	4

Propriétés des ressources

Constante	Type	Valeur
Bit ressource heap système	Entier long	6
Bit ressource modifiée	Entier long	1
Bit ressource préchargée	Entier long	2
Bit ressource protégée	Entier long	3
Bit ressource purgeable	Entier long	5
Bit ressource verrouillée	Entier long	4
Masque ressource heap système	Entier long	64
Masque ressource modifiée	Entier long	2
Masque ressource préchargée	Entier long	4
Masque ressource protégée	Entier long	8
Masque ressource purgeable	Entier long	32
Masque ressource verrouillée	Entier long	16

Propriétés plate-forme

Constante	Type	Valeur
Autres G3 et supérieurs	Entier long	406
INTEL 386	Entier long	386
INTEL 486	Entier long	486
Macintosh 68K	Entier long	1
Pentium	Entier long	586
Power Macintosh	Entier long	2
PowerPC 601	Entier long	601
PowerPC 603	Entier long	603
PowerPC 604	Entier long	604
PowerPC G3	Entier long	510
Windows	Entier long	3

Signatures système standard

Constante	Type	Valeur
Document image	Chaîne	PICT
Document MIDI Windows	Chaîne	MID
Document son Windows	Chaîne	WAV
Document texte	Chaîne	TEXT
Document vidéo Windows	Chaîne	AVI
QT Compresseur animation	Chaîne	rle
QT Compresseur Compact Vidéo	Chaîne	cdvc
QT Compresseur graphiques	Chaîne	smc
QT Compresseur photo	Chaîne	jpeg
QT Compresseur RAW	Chaîne	raw
QT Compresseur vidéo	Chaîne	rpza

Statut du process

Constante	Type	Valeur
Détruit	Entier long	-1
Dialogue caché	Entier long	6
En attente drapeau interne	Entier long	4
En attente entrée sortie	Entier long	3
En attente événement	Entier long	2
En exécution	Entier long	0
Endormi	Entier long	1
Inexistant	Entier long	-100
Suspendu	Entier long	5

Styles de caractères

Constante	Type	Valeur
Condensé	Entier long	32
Etendu	Entier long	64
Gras	Entier long	1
Italique	Entier long	2
Normal	Entier long	0
Ombré	Entier long	16
Relief	Entier long	8
Souligné	Entier long	4

Touches de fonction

Constante	Type	Valeur
Touche aide	Entier long	5
Touche bas	Entier long	31
Touche début	Entier long	1
Touche droite	Entier long	29
Touche échappement	Entier long	27
Touche entrée	Entier long	3
Touche F1	Entier long	-122
Touche F10	Entier long	-109
Touche F11	Entier long	-103
Touche F12	Entier long	-111
Touche F13	Entier long	-105
Touche F14	Entier long	-107
Touche F15	Entier long	-113
Touche F2	Entier long	-120
Touche F3	Entier long	-99
Touche F4	Entier long	-118
Touche F5	Entier long	-96
Touche F6	Entier long	-97
Touche F7	Entier long	-98
Touche F8	Entier long	-100
Touche F9	Entier long	-101
Touche fin	Entier long	4
Touche gauche	Entier long	28
Touche haut	Entier long	30
Touche page précédente	Entier long	12
Touche page suivante	Entier long	11
Touche retour arrière	Entier long	8
Touche retour chariot	Entier long	13
Touche tab	Entier long	9

Type de process

Constante	Type	Valeur
Aucun	Entier long	0
Autre process 4D	Entier long	-10
Autre process utilisateur	Entier long	4
Créé depuis le mode Utilisation	Entier long	3
Créé par commande de menu	Entier long	2
Créé par programmation	Entier long	1
Gestionnaire Apple Event	Entier long	-7
Gestionnaire d'événement	Entier long	-8
Gestionnaire d'index	Entier long	-5
Gestionnaire du cache	Entier long	-4
Gestionnaire du port série	Entier long	-6
Process de structure	Entier long	-2
Process principal	Entier long	-1
Process Web avec contexte	Entier long	-11
Process Web sans contexte	Entier long	-3
Tâche externe	Entier long	-9

Type fenetre

Constante	Type	Valeur
Fenêtre externe	Entier long	5
Fenêtre flottante	Entier long	14
Fenêtre modale	Entier long	9
Fenêtre normale	Entier long	8

Types champs et variables

Constante	Type	Valeur
Est un BLOB	Entier long	30
Est un booléen	Entier long	6
Est un champ alpha	Entier long	0
Est un entier	Entier long	8
Est un entier long	Entier long	9
Est un numérique	Entier long	1
Est un pointeur	Entier long	23
Est un tableau 2D	Entier long	13
Est un tableau booléen	Entier long	22
Est un tableau chaîne	Entier long	21
Est un tableau date	Entier long	17
Est un tableau entier	Entier long	15
Est un tableau entiers long	Entier long	16
Est un tableau image	Entier long	19
Est un tableau numérique	Entier long	14
Est un tableau pointeur	Entier long	20
Est un tableau texte	Entier long	18
Est un texte	Entier long	2
Est une date	Entier long	4
Est une heure	Entier long	11
Est une image	Entier long	3
Est une sous table	Entier long	7
Est une variable chaîne	Entier long	24
Est une variable indéfinie	Entier long	5

Index des commandes

A

Abs 617
Activation 565
ACTIVER BOUTON 1107
ACTIVER LIGNE MENU 943
ADJOINDRE ELEMENT 472
AFFICHER BARRE DE MENUS 926
AFFICHER BARRE OUTILS 602
AFFICHER ENREGISTREMENT 425
AFFICHER FENETRE 594
Ajouter a date 296
Ajouter a document 388
AJOUTER A LISTE 891
AJOUTER A PRESSE PAPIERS 1023
AJOUTER ENREGISTREMENT 1231
AJOUTER LIGNE MENU 944
AJOUTER SEGMENT DE DONNEES 509
AJOUTER SOUS ENREGISTREMENT 1236
ALERTE 953
ALLER A CHAMP 664
ALLER A DERNIER ENREGISTREMENT 1256
ALLER A DERNIER SOUS ENREGISTREMENT 1448
ALLER A ENREGISTREMENT 436
ALLER A PAGE 1018
ALLER DANS SELECTION 1252
Ancien 1243
ANCIEN LIEN RETOUR 862
Annee de 293
ANNULER TRANSACTION 1546
Appartient a ensemble 475
Appartient au groupe 1574
Appel exterieur 567
APPELER 4D 283
APPELER PROCESS 1082
APPELER SUR A PROPOS 927
APPELER SUR ERREUR 816
APPELER SUR EVENEMENT 811
Application compilee 495

APPLIQUER A SELECTION 1267
APPLIQUER A SOUS SELECTION 1446
Apres 561
Arctan 630
ARRETER SERVEUR WEB 1411
Arrondi 620
ASSOCIER TYPES FICHIER 408
Avant 559
Avant selection 1258
Avant sous enregistrement 1451

B

BEEP 775
BLOB VERS DOCUMENT 156
BLOB vers entier 179
BLOB vers entier long 181
BLOB VERS IMAGE 712
BLOB vers liste 165
BLOB vers reel 183
BLOB vers texte 185
BLOB VERS VARIABLE 161

C

CACHER BARRE DE MENUS 925
CACHER BARRE OUTILS 601
CACHER FENETRE 593
CACHER PROCESS 1095
Caractere 216
Chaine 205
Chaine heure 299
Champ 356
CHANGER BARRE 923
CHANGER COORDONNEES FENETRE 607
CHANGER CREATEUR DOCUMENT 382
CHANGER ELEMENT 910
CHANGER JEU DE CARACTERES 1103

CHANGER LICENCES 1590
CHANGER MOT DE PASSE 1572
CHANGER POINTEUR SOURIS 804
CHANGER POSITION DANS DOCUMENT 421
CHANGER PRIVILEGES 1569
CHANGER PROPRIETES DOCUMENT 416
CHANGER PROPRIETES ELEMENT 900
CHANGER PROPRIETES LISTE 878
CHANGER RACCOURCI CLAVIER 941
CHANGER STYLE 1105
CHANGER STYLE LIGNE MENU 937
CHANGER TAILLE 1104
CHANGER TAILLE DOCUMENT 419
CHANGER TEXTE LIGNE MENU 934
CHANGER TITRE FENETRE 599
CHANGER TYPE DOCUMENT 380
CHANGER UTILISATEUR 1570
CHARGER ANCIEN 860
CHARGER ENREGISTREMENT 457
CHARGER ENSEMBLE 479
CHARGER ET COMPRESSER IMAGE 705
Charger liste 867
CHARGER SUR LIEN 851
CHERCHER 1144
CHERCHER DANS SELECTION 1151
Chercher dans tableau 1517
Chercher fenetre 610
CHERCHER PAR EXEMPLE 1143
CHERCHER PAR FORMULE 1153
CHERCHER PAR FORMULE DANS SELECTION 1155
CHERCHER PAR TABLEAU 1156
Chercher process 1063
CHERCHER SOUS ENREGISTREMENTS 1454
CHERCHER SUR CLE 237
CHOIX COULEUR 1123
CHOIX ENUMERATION 1118
CHOIX FILTRE SAISIE 1116
CHOIX FORMATAGE 1113
CHOIX SAISISSABLE 1119
CHOIX VISIBLE 1121

Code ascii 214
COMPRESSER BLOB 148
COMPRESSER FICHIER IMAGE 707
COMPRESSER IMAGE 704
CONFIRMER 955
Connexion Web securisee 1437
Contexte Web 1428
Convertisseur Euro 634
COORDONNEES ECRAN 522
COORDONNEES FENETRE 606
COPIER BLOB 190
COPIER DOCUMENT 390
COPIER ENSEMBLE 487
Copier liste 871
COPIER SELECTION 1279
COPIER TABLEAU 1521
Cos 628
Createur document 381
CREER ALIAS 399
Creer document 386
CREER DOSSIER 394
CREER ENREGISTREMENT 426
CREER ENSEMBLE SUR TABLEAU 470
Creer fenetre 582
Creer fenetre externe 586
Creer fenetre formulaire 611
Creer fichier ressources 1194
CREER IMAGETTE 719
CREER SELECTION SUR TABLEAU 1284
CREER SOUS ENREGISTREMENT 1441
CREER SUR LIEN 858
CRYPTER BLOB 191
CUMULER SUR 758
C_ALPHA 280
C_BLOB 271
C_BOOLEEN 272
C_DATE 273
C_ENTIER 275
C_ENTIER LONG 276
C_GRAPHE 274

C_HEURE 282
C_IMAGE 277
C_POINTEUR 278
C_REEL 279
C_TEXTE 281

D

Date 297
Date du jour 287
DEBUT SELECTION 1254
DEBUT SOUS ENREGISTREMENT 1447
DEBUT TRANSACTION 1544
Dec 619
DECOMPRESSER BLOB 150
DECRYPTER BLOB 196
Demander 958
DEPILER ENREGISTREMENT 443
DEPLACER DOCUMENT 391
DEPLACER FENETRE 591
DEPLACER OBJET 1130
DEPLACER SELECTION 1281
Dernier objet 805
DERNIERE PAGE 1015
Desactivation 566
DESINSCRIRE CLIENT 1073
DIALOGUE 1239
DIFFERENCE 481
DOCUMENT VERS BLOB 154
Dossier 4D 505
Dossier systeme 531
Dossier temporaire 533
DUPLIQUER ENREGISTREMENT 427

E

Ecart type 644
Ecran principal 526

ECRIRE CACHE 510
ECRIRE FICHIER IMAGE 713
ECRIRE IMAGE DANS BIBLIOTHEQUE 725
ECRIRE IMAGE DANS PRESSE PAPIERS 1035
ECRIRE NOM RESSOURCE 1220
Ecrire proprietes groupe 1587
ECRIRE PROPRIETES RESSOURCE 1222
Ecrire proprietes utilisateur 1581
ECRIRE RESSOURCE 1215
ECRIRE RESSOURCE CHAINE 1206
ECRIRE RESSOURCE IMAGE 1210
ECRIRE RESSOURCE TEXTE 1208
ECRIRE TEXTE DANS PRESSE PAPIERS 1036
ECRIRE VARIABLE PROCESS 1087
ECRIRE VARIABLES 1595
ECRITURE ASCII 733
ECRITURE DIF 741
ECRITURE SYLK 737
EFFACER ENSEMBLE 474
EFFACER FENETRE 589
EFFACER PRESSE PAPIERS 1029
EFFACER SELECTION 1283
EFFACER SEMAPHORE 1080
EFFACER VARIABLE 1598
Element parent 904
Element selectionne 912
EMPILER ENREGISTREMENT 442
En entete 562
En pied 564
En rupture 563
ENDORMIR PROCESS 1053
ENLEVER ELEMENT 473
Enregistrement charge 430
Enregistrement modifie 429
ENREGISTREMENT PRECEDENT 1257
ENREGISTREMENT SELECTION 1272
ENREGISTREMENT SUIVANT 1255
Enregistrement verrouille 459
Enregistrements dans ensemble 476
Enregistrements dans table 434

Enregistrements trouves 1248
ENREGISTRER EVENEMENT 537
ENREGISTRER IMAGE 708
ENSEMBLE VIDE 468
Ent 618
ENTIER LONG VERS BLOB 170
ENTIER VERS BLOB 167
ENUMERATION VERS TABLEAU 1522
ENVOYER BLOB HTML 1422
ENVOYER ENREGISTREMENT 259
ENVOYER FICHIER HTML 1418
ENVOYER PAQUET 250
ENVOYER REDIRECTION HTTP 1434
ENVOYER TEXTE HTML 1425
ENVOYER VARIABLE 257
Est une variable 834
ETAT 749
Etat lecture seulement 456
Evenement formulaire 541
Evenement moteur 1562
EXECUTER 836
EXECUTER SUR CLIENT 1068
Executer sur serveur 1048
Exp 626
EXPORTER DONNEES 745

F

Faux 201
Fenetre premier plan 608
Fenetre suivante 609
FERMER DOCUMENT 389
FERMER FENETRE 588
FERMER FICHIER RESSOURCES 1196
Fichier application 500
Fichier donnees 503
Fichier structure 501
FILTRER EVENEMENT 815
FILTRER FRAPPE CLAVIER 658

Fin de selection 1260
Fin sous enregistrement 1452
FIXER COULEURS RVB 1125
FIXER DESTINATION RECHERCHE 1157
FIXER ENTETE HTTP 1429
FIXER HISTORIQUE 513
FIXER INDEX 361
FIXER INTERFACE 779
FIXER LIMITE RECHERCHE 1163
FIXER LIMITES AFFICHAGE WEB 1414
FIXER MINUTEUR 570
FIXER NIVEAU COMPARAISON REEL 631
FIXER PAGE ACCUEIL 1417
FIXER PARAMETRE BASE 365
FIXER PROFONDEUR ECRAN 525
FIXER RACINE HTML 1413
FIXER TAILLE BLOB 146
FIXER TEMPORISATION WEB 1412
FIXER TIMEOUT 246
FIXER TITRES CHAMPS 785
FIXER TITRES TABLES 781
FORMULAIRE ENTREE 1460
FORMULAIRE SORTIE 1462
Frappe clavier 653

G

GENERER CLES CRYPTAGE 1135
GENERER CLIC 799
GENERER DEMANDE CERTIFICAT 1137
GENERER EVENEMENT 800
GENERER FRAPPE CLAVIER 798
Gestalt 536
GRAPHE 689
GRAPHE SUR SELECTION 696

H

Hasard 622
Hauteur barre de menus 527
Hauteur ecran 519
Heure 300
Heure courante 298

I

IMAGE VERS BLOB 711
IMAGE VERS GIF 709
IMPORTER DONNEES 743
IMPRESSION ECRAN 769
IMPRIMER ENREGISTREMENT 764
IMPRIMER ETIQUETTES 751
IMPRIMER LIGNE 770
IMPRIMER SELECTION 754
INACTIVER BOUTON 1109
INACTIVER LIGNE MENU 942
Indefinie 1600
INFORMATION ELEMENT 908
INFORMATIONS PROCESS 1060
INSCRIRE CLIENT 1070
Inserer chaine 223
INSERER DANS BLOB 188
INSERER ELEMENT 899
INSERER LIGNE MENU 946
INSERER LIGNES 1519
INTERSECTION 483
INVERSER FOND 807
ISO vers Mac 232

J

JOINTURE 863
JOUER SON 776

Jour de 290

L

LAISSER MESSAGES 952
LANCER SERVEUR WEB 1410
Largeur ecran 520
LECTURE ASCII 731
LECTURE DIF 739
LECTURE ECRITURE 454
LECTURE SEULEMENT 455
LECTURE SYLK 735
LIBERER ENREGISTREMENT 458
LIEN RETOUR 855
LIENS AUTOMATIQUES 850
Lire chaine dans liste 1204
LIRE CLIENTS INSCRITS 1074
LIRE ENTETE HTTP 1431
LIRE FICHIER IMAGE 714
LIRE ICONE DOCUMENT 417
Lire ID ressource composant 1227
LIRE IMAGE DANS BIBLIOTHEQUE 723
LIRE IMAGE DANS PRESSE PAPIERS 1032
LIRE INFORMATIONS SERIALISATION 515
Lire interface 778
LIRE LISTE GROUPE 1584
LIRE LISTE UTILISATEURS 1577
Lire nom ressource 1218
Lire parametre base 363
LIRE PRESSE PAPIERS 1030
LIRE PROPRIETES BLOB 152
LIRE PROPRIETES CHAMP 357
LIRE PROPRIETES ELEMENT 902
LIRE PROPRIETES FORMULAIRE 613
LIRE PROPRIETES GROUPE 1585
LIRE PROPRIETES LIEN 360
LIRE PROPRIETES LISTE 887
Lire proprietes ressource 1221
LIRE PROPRIETES SAISIE CHAMP 359

LIRE PROPRIETES TABLE 355
LIRE PROPRIETES UTILISATEUR 1579
LIRE RECT OBJET 1129
LIRE RESSOURCE 1213
Lire ressource chaine 1205
LIRE RESSOURCE ICONE 1211
LIRE RESSOURCE IMAGE 1209
Lire ressource texte 1207
Lire texte dans presse papiers 1033
Lire texte edite 568
LIRE VARIABLE PROCESS 1084
LIRE VARIABLES 1597
LIRE VARIABLES FORMULAIRE WEB 1426
LISTE DE CHAINES VERS TABLEAU 1201
LISTE DES DOCUMENTS 407
LISTE DES DOSSIERS 406
LISTE DES POLICES 528
LISTE DES VOLUMES 402
Liste existante 876
LISTE FENETRES 603
LISTE IMAGES DANS BIBLIOTHEQUE 721
LISTE RESSOURCES 1199
LISTE SEGMENTS DE DONNEES 507
LISTE TYPES IMAGES 715
LISTE TYPES RESSOURCE 1197
LISTE VERS BLOB 163
Log 625
Longueur 213

M

Mac vers ISO 229
Mac vers Windows 227
Macintosh commande enfoncee 791
Macintosh control enfoncee 793
Macintosh option enfoncee 792
Majusc 220
Majuscule enfoncee 787
Marque ligne menu 938

MARQUER ENREGISTREMENTS 1273
MARQUER LIGNE MENU 939
Max 643
MAXIMISER FENETRE 595
Menu choisi 928
MESSAGE 961
Min 642
MINIMISER FENETRE 597
Minusc 221
Modifie 1241
MODIFIER ENREGISTREMENT 1234
MODIFIER SELECTION 1266
MODIFIER SOUS ENREGISTREMENT 1238
Modulo 623
Mois de 291
MONTRER PROCESS 1096
Moyenne 641

N

NE PAS VALIDER 652
Nil 833
Niveau 762
Niveau du trigger 1564
NIVEAUX DE RUPTURES 757
Nom commande 837
Nom de la machine 534
Nom de la table 352
Nom de police 529
Nom du champ 353
Nom du possesseur 535
Nom methode courante 840
Nombre de champs 351
Nombre de lignes du menu 931
Nombre de menus 930
Nombre de millisecondes 302
Nombre de parametres 825
Nombre de process 1066
Nombre de process utilisateurs 1067

Nombre de tables 350
Nombre de ticks 301
Nombre ecrans 521
Nombre elements 874
Nombre utilisateurs 1065
NOMMER ENSEMBLE 469
Non 202
Nouveau process 1045
Nouvel enregistrement 428
Nouvelle liste 870
Num 208
Numero dans selection 1251
Numero de police 530
Numero du jour 294
Numero du process courant 1057
Numero enregistrement 435
Numerotation automatique 437

O

Ouvrir document 383
Ouvrir fichier ressources 1190
OUVRIR URL WEB 1438

P

Page formulaire courante 1019
Page impression 756
PAGE PRECEDENTE 1017
PAGE SUIVANTE 1016
PARAMETRES DU GRAPHE 694
PARAMETRES IMPRESSION 768
PAS DE TRACE 843
PASSER AU PREMIER PLAN 1097
Pendant 560
Pointeur vers 835
Pop up menu 795
Position 210

Position dans document 420
Position deposer 678
Position element liste 903
POSITION MESSAGE 966
POSITION SOURIS 794
PREMIERE PAGE 1014
Process de la fenetre 605
Process de premier plan 1098
Process interrompu 1056
PROFONDEUR ECRAN 523
PROPRIETES DOCUMENT 410
PROPRIETES DU TRIGGER 1565
PROPRIETES DU VOLUME 403
PROPRIETES GLISSER DEPOSER 679
PROPRIETES IMAGE 718
PROPRIETES PLATE FORME 496

Q

QUITTER 4D 511

R

Raccourci clavier 940
Racine carree 624
REACTIVER PROCESS 1055
RECEVOIR BUFFER 255
RECEVOIR ENREGISTREMENT 260
RECEVOIR PAQUET 252
RECEVOIR VARIABLE 258
REDESSINER 806
REDESSINER FENETRE 590
REDESSINER LISTE 877
REDUIRE SELECTION 1269
REEL VERS BLOB 173
REFUSER 665
REGLER SERIE 241
Remplacer caracteres 222

Remplacer chaine 225
RESOUDRE ALIAS 401
RESOUDRE POINTEUR 831
REUNION 485

S

SAUT DE PAGE 772
SCAN INDEX 1271
SELECTION LIMITEE VERS TABLEAU 1528
SELECTION RETOUR 864
SELECTION VERS TABLEAU 1526
Selectionner dossier 395
SELECTIONNER ELEMENT 914
SELECTIONNER ELEMENT PAR REFERENCE 916
SELECTIONNER TEXTE 803
Self 830
Semaphore 1077
SIECLE PAR DEFAUT 303
Sin 627
Somme 640
Somme des carres 647
Sous chaine 211
SOUS ENREGISTREMENT PRECEDENT 1450
SOUS ENREGISTREMENT SUIVANT 1449
Sous enregistrements trouves 1445
Sous total 759
STATISTIQUES DU CACHE WEB 1436
Statut du process 1058
STOCKER ANCIEN 861
STOCKER ENREGISTREMENT 431
STOCKER ENSEMBLE 477
STOCKER LISTE 869
STOCKER SUR LIEN 859
STOP 820
Style ligne menu 935
Supprimer chaine 224
SUPPRIMER DANS BLOB 189
SUPPRIMER DOCUMENT 392

SUPPRIMER DOSSIER 398
SUPPRIMER ELEMENT 906
SUPPRIMER ENREGISTREMENT 433
SUPPRIMER IMAGE DANS BIBLIOTHEQUE 728
SUPPRIMER LIGNE MENU 947
SUPPRIMER LIGNES 1520
SUPPRIMER LISTE 872
SUPPRIMER MESSAGES 951
SUPPRIMER RESSOURCE 1225
SUPPRIMER SELECTION 1249
SUPPRIMER SOUS ENREGISTREMENT 1442
SUPPRIMER UTILISATEUR 1575
SUSPENDRE PROCESS 1054

T

Table 354
Table du formulaire courant 1464
TABLE PAR DEFAUT 1457
Table par defaut courante 1459
TABLEAU ALPHA 1502
TABLEAU BOOLEEN 1508
TABLEAU BOOLEEN SUR ENSEMBLE 1536
TABLEAU DATE 1506
TABLEAU ENTIER 1496
TABLEAU ENTIER LONG 1498
TABLEAU ENTIER LONG SUR SELECTION 1535
TABLEAU IMAGE 1510
TABLEAU POINTEUR 1512
TABLEAU REEL 1500
TABLEAU TEXTE 1504
TABLEAU VERS ENUMERATION 1524
TABLEAU VERS LISTE DE CHAINES 1202
TABLEAU VERS SELECTION 1531
Taille BLOB 147
Taille document 418
Taille image 717
Taille tableau 1514
Tan 629

Tester chemin acces 393
Tester presse papiers 1037
Tester semaphore 1081
Texte ligne menu 933
TEXTE SELECTIONNE 802
TEXTE VERS BLOB 176
TITRE BOUTON 1111
Titre fenetre 598
Titre menu 932
TOUS LES SOUS ENREGISTREMENTS 1444
TOUT SELECTIONNER 1247
TRACE 841
Transaction en cours 1547
TRIER 1166
TRIER LISTE 889
TRIER PAR FORMULE 1171
TRIER SOUS ENREGISTREMENTS 1453
TRIER SUR INDEX 238
TRIER TABLEAU 1515
Troncature 621
Trouver clef index 1165
Type 827
Type application 491
Type document 379
Type fenetre 604
Type version 492

U

Utilisateur courant 1573
Utilisateur supprime 1576
UTILISER ENSEMBLE 471
UTILISER FILTRE 248
UTILISER PARAMETRES IMPRESSION 766
UTILISER SELECTION 1282

V

VALEURS DISTINCTES 1533
VALIDER 651
Valider mot de passe 1571
VALIDER TRANSACTION 1545
VARIABLE VERS BLOB 158
VARIABLE VERS VARIABLE 1090
Variance 645
Verrouillage majuscule enfoncee 788
VERROUILLE PAR 460
Version application 493
VISUALISER SELECTION 1262
Vrai 200

W

Windows Alt enfoncee 790
Windows Ctrl enfoncee 789
Windows vers Mac 228